CS402 ADA – Unit 2 easy and detailed notes

Greedy Strategy Notes

(RGPV Exam-Oriented, Easy Hinglish)

A. Introduction

Greedy Strategy ek algorithm design technique hai jisme hum har step par **currently best choice** select karte hain.

Simple meaning:

> Jo option abhi sabse best lag raha hai, use choose karo.

Real-life example:

Agar tumhare paas ₹100 hain aur tum maximum chocolates lena chahte ho, to tum sabse cheap chocolate pehle choose karoge. Ye greedy thinking hai.

B. Definition

Exam Definition

Greedy Method is an algorithm design technique in which a solution is constructed step by step by choosing the locally optimal choice at each stage, with the hope of obtaining a globally optimal solution.

Easy Explanation

Greedy algorithm har step par best local decision leta hai.

Local best ka matlab:

Abhi ke moment par jo sabse profitable / smallest / largest / minimum cost / maximum benefit choice ho.

C. Core Concept

Greedy strategy ke 2 main properties hote hain:

1. Greedy Choice Property

Iska matlab:

A global optimal solution can be reached by choosing a local optimal choice.

Easy:

Har step ka best decision final best answer tak le ja sakta hai.

Example:

Fractional Knapsack mein highest profit/weight ratio wala item pehle lena best hota hai.

---

2. Optimal Substructure

Iska matlab:

Problem ka optimal solution uske subproblems ke optimal solutions se banta hai.

Easy:

Chhote-chhote best answers milkar final best answer banate hain.

Important Point

Greedy har problem ke liye kaam nahi karta.

Greedy works only when:

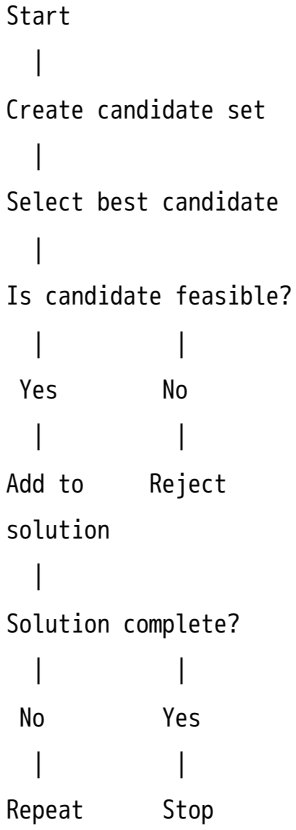
1. Greedy choice property exists.
2. Optimal substructure exists.

D. Working / Procedure

General Steps of Greedy Algorithm

1. Candidate set banao.
2. Best candidate select karo.
3. Check karo candidate feasible hai ya nahi.
4. Agar feasible hai to solution mein add karo.
5. Repeat until solution complete ho jaye.

Flowchart



General Pseudocode

```
Greedy(C)

S = empty solution

while solution is not complete:

    x = best candidate from C

    remove x from C

    if x is feasible:

        add x to S

return S
```

E. Dry Run Example

Problem

Select maximum number of activities without overlap.

Activities:

Activity	Start	Finish
A1	1	3
A2	2	5
A3	4	6
A4	6	8

Greedy Rule

Choose activity with earliest finish time.

Step 1

Sort by finish time:

A1, A2, A3, A4

Step 2

Select A1 because finish time is 3.

Selected:

```
A1
```

Step 3

A2 starts at 2, but A1 finishes at 3.

A2 overlaps, so reject.

Step 4

A3 starts at 4, after A1 finishes.

Select A3.

Selected:

```
A1, A3
```

Step 5

A4 starts at 6, after A3 finishes.

Select A4.

Selected:

```
A1, A3, A4
```

Final Answer

Maximum non-overlapping activities:

`A1, A3, A4`

F. Complexity Analysis

Greedy algorithm ki complexity problem par depend karti hai.

Usually greedy algorithm mein:

1. Sorting hoti hai.
2. Selection hota hai.

Sorting Cost

`O(n log n)`

Selection Cost

`O(n)`

Total Complexity

`O(n log n)`

Space Complexity

Usually:

`O(1)` or `O(n)`

Depends on storage used.

Why O(n log n)?

Because many greedy algorithms first sort candidates.

Example:

- Fractional Knapsack: sort by profit/weight ratio
- Job Sequencing: sort by profit
- Kruskal: sort edges by weight
- Activity Selection: sort by finish time

Sorting dominates total time.

G. Advantages

1. Simple to understand.
2. Easy to implement.
3. Faster than Dynamic Programming.
4. Uses less memory.
5. Works well for optimization problems.

H. Disadvantages

1. Does not always give optimal answer.
2. Cannot be used for every problem.
3. Decision once taken is not changed.
4. Requires greedy choice property.

I. Applications

Greedy strategy is used in:

1. Huffman Coding
2. Fractional Knapsack
3. Job Sequencing with Deadlines
4. Prim's Algorithm
5. Kruskal's Algorithm

6. Dijkstra Algorithm
7. Optimal Merge Pattern

J. Common Mistakes

❌ Greedy always gives optimal solution.
✅ No, only for problems with greedy choice property.

❌ Greedy and Dynamic Programming are same.
✅ No. Greedy makes immediate choice, DP stores subproblem results.

❌ Greedy checks all possibilities.
✅ No. Greedy chooses one best option at each step.

K. Viva Questions

Q1. What is Greedy Method?

Greedy Method is an algorithm design technique that chooses the best option at each step.

Q2. What is Greedy Choice Property?

It means local optimal choices can lead to global optimal solution.

Q3. What is Optimal Substructure?

It means optimal solution contains optimal solutions of subproblems.

Q4. Give examples of Greedy Algorithms.

Huffman Coding, Prim, Kruskal, Dijkstra, Fractional Knapsack.

Q5. Does Greedy always give optimal solution?

No.

L. Exam Keywords

Write these words in exam:

- Locally optimal choice
- Globally optimal solution
- Greedy choice property
- Optimal substructure
- Feasible solution
- Candidate set
- Optimization problem
- Fast execution
- No backtracking
- Step-by-step selection

M. Memory Trick

Greedy = “Abhi ka Best”

Mnemonic:

C-S-F-A

Candidate set

Select best

Feasibility check

Add to solution

N. Comparison Table

Greedy vs Dynamic Programming

Basis	Greedy	Dynamic Programming
Approach	Local best choice	Solves all subproblems
Decision	Once taken, not changed	Previous results stored
Speed	Faster	Slower
Memory	Less	More
Optimality	Not always guaranteed	Usually guaranteed
Example	Fractional Knapsack	0/1 Knapsack

O. RGPV Exam Answers

2 Mark Answer

Greedy Method is an algorithm design technique in which the solution is built step by step by choosing the locally optimal choice at each stage. It is used in optimization problems.

---

5 Mark Answer

Greedy Method is a technique used to solve optimization problems. In this method, the algorithm makes the best possible choice at each step without reconsidering previous choices.

It works when the problem has:

1. Greedy Choice Property
2. Optimal Substructure

Examples:

- Huffman Coding
- Prim's Algorithm
- Kruskal's Algorithm
- Dijkstra Algorithm
- Fractional Knapsack

Greedy algorithms are simple, fast, and memory efficient.

---

7 Mark Answer

Greedy Method is an algorithm design technique that builds a solution step by step. At each step, it selects the best available option, called the locally optimal choice.

The main idea is that local optimal choices may lead to a global optimal solution.

Greedy algorithm works on two properties:

1. Greedy Choice Property
2. Optimal Substructure

General steps:

1. Create candidate set.
2. Select best candidate.
3. Check feasibility.
4. Add to solution.
5. Repeat until solution is complete.

Examples:

- Huffman Coding
- Prim's Algorithm
- Kruskal's Algorithm
- Dijkstra Algorithm

Advantages:

- Simple
- Fast
- Efficient

Disadvantage:

- Does not always give optimal solution.

---

10 Mark Topper Answer

Greedy Method is an important algorithm design strategy used mainly for optimization problems. It constructs the solution step by step by selecting the locally optimal choice at each stage.

A greedy algorithm never revises its previous decisions. It assumes that choosing the best option at the current step will lead to the best overall solution.

The two important properties of greedy algorithms are:

1. **Greedy Choice Property** – A globally optimal solution can be obtained by making locally optimal choices.
2. **Optimal Substructure** – The optimal solution of a problem contains optimal solutions of its subproblems.

General Algorithm:

```
S = empty solution

while solution is not complete:
    select best candidate
    if candidate is feasible:
        add candidate to solution

return S
```

Flow:

Candidate Set → Best Selection → Feasibility Check → Add to Solution

Applications:

- Huffman Coding
- Fractional Knapsack
- Job Sequencing
- Prim's Algorithm
- Kruskal's Algorithm
- Dijkstra Algorithm

Advantages:

- Easy to implement
- Fast execution
- Less memory usage

Limitations:

- It does not always give optimal solution.
- It works only when greedy choice property and optimal substructure are present.

Conclusion:

Greedy Method is a powerful and efficient technique for solving many optimization problems where local decisions lead to global optimum.

P. One Page Revision Sheet

Greedy Strategy

Definition:
Choose the best option at each step.

Main Idea

Local Best → Global Best

Properties

1. Greedy Choice Property
2. Optimal Substructure

General Steps

Candidate Set
↓
Select Best Candidate

↓

Check Feasibility

↓

Add to Solution

↓

Repeat

Pseudocode

```
S = empty

while not complete:
    x = best candidate
    if feasible:
        add x to S

return S
```

Complexity

Usually:

`O(n log n)` because sorting is often required.

Examples

- Huffman Coding
- Fractional Knapsack
- Job Sequencing
- Prim
- Kruskal
- Dijkstra

Keywords

Locally optimal choice, globally optimal solution, feasible solution, optimal substructure, candidate set.

Memory Trick

C-S-F-A

Candidate → Select → Feasible → Add

Optimal Merge Pattern (OMP)

(RGPV Exam-Oriented Complete Notes | Easy Hinglish)

A. Introduction

Suppose tumhare paas 4 sorted files hain:

```
File 1 = 10 records
File 2 = 20 records
File 3 = 30 records
File 4 = 40 records
```

Ab in sabko merge karke ek final file banani hai.

Question:

Kis order mein merge karein taki total cost minimum ho?

Yahi problem **Optimal Merge Pattern (OMP)** solve karta hai.

Real-Life Analogy

Imagine:

Tumhare paas 4 books hain:

10 pages, 20 pages, 30 pages, 40 pages

Agar tum sabse badi books pehle merge karoge to zyada work lagega.

Greedy idea:

👉 Hamesha sabse chhoti 2 files ko pehle merge karo.

B. Definition

Exam Definition

Optimal Merge Pattern is a Greedy algorithm used to merge multiple files with minimum total merging cost by repeatedly combining the two smallest files.

Easy Explanation

OMP ka rule:

"Always merge the two smallest files first."

Why?

Small files merge karne ki cost kam hoti hai.

Agar hum pehle large files merge karenge to total cost badh jayegi.

C. Core Concept

Keywords

File

Data ka collection.

Merge

Do files ko combine karna.

Cost

Merge cost = Sum of file sizes.

Example:

```
10 + 20 = 30
```

Cost = 30

Greedy Choice Property

Greedy rule:

Always choose smallest two files.

Optimal Substructure

Small optimal merges milkar final optimal solution banate hain.

Why Greedy Works?

Because:

Small files ko pehle merge karne se future merge costs bhi kam rehte hain.

D. Working

General Steps

Step 1:

Insert all file sizes.

Step 2:

Pick smallest two files.

Step 3:

Merge them.

Step 4:

Add merge cost to total cost.

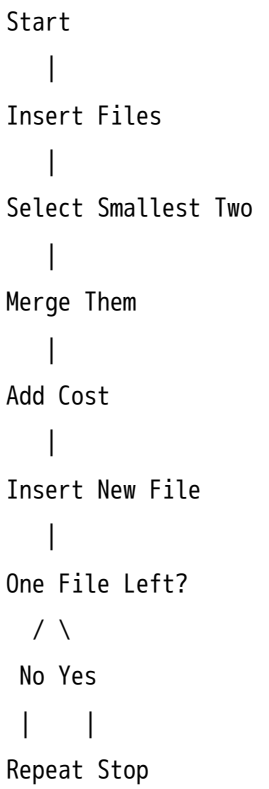
Step 5:

Insert merged file back.

Step 6:

Repeat until only one file remains.

Flowchart



Pseudocode

```
OMP(files)

cost = 0

while more than one file exists

   x = smallest file

   y = second smallest file

   merge = x + y
```

```
    cost += merge

    insert merge

return cost
```

E. Dry Run (Most Important)

Example

Files:

```
10 20 30 40
```

Iteration 1

Pick smallest:

```
10 and 20
```

Merge:

```
30
```

Cost:

```
30
```

Remaining:

```
30 30 40
```

Iteration 2

Pick:

```
30 and 30
```

Merge:

```
60
```

Cost:

```
30 + 60 = 90
```

Remaining:

```
40 60
```

Iteration 3

Pick:

```
40 and 60
```

Merge:

```
100
```

Cost:

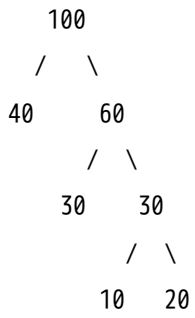
```
90 + 100
= 190
```

Final Answer

Total Minimum Cost:

```
190
```

Tree Representation



F. Complexity Analysis

To efficiently select smallest files:

Use Min Heap / Priority Queue.

Best Case

O(nlogn)

Average Case

O(nlogn)

Worst Case

O(nlogn)

Space Complexity

O(n)

Why O(n log n)?

Each extraction and insertion in heap:

```
O(log n)
```

Performed approximately n times.

Therefore:

```
O(n log n)
```

G. Advantages

1

Minimum merging cost.

2

Simple Greedy solution.

3

Efficient for large files.

4

Used in external sorting.

H. Disadvantages

1

Works only when greedy property exists.

2

Not suitable for every optimization problem.

I. Applications

External Sorting

Large database files.

File Compression

Huffman Coding.

Database Management Systems

Data merging.

Data Processing

Combining sorted datasets.

J. Common Mistakes

❌ Merge largest files first.

✔ Always merge smallest files first.

❌ Forgeting to insert merged file back.

✔ Always reinsert merged file.

❌ Adding only final merge cost.

✔ Add every merge cost.

K. Viva Questions

What is Optimal Merge Pattern?

Greedy algorithm for minimum merge cost.

What is merge cost?

Sum of file sizes being merged.

Which files are merged first?

Smallest two files.

Which data structure is used?

Min Heap / Priority Queue.

Technique used?

Greedy Method.

L. Exam Keywords

Write these keywords:

- Greedy Strategy
- Minimum Cost
- Merge Operation
- Priority Queue
- Min Heap
- Smallest Files
- Optimal Solution
- External Sorting
- Huffman Coding

M. Memory Trick

OMP Rule

"Small + Small First"

Remember:

Small

-

Small

First

N. Comparison Table

Optimal Merge vs Huffman Coding

Feature	OMP	Huffman
Technique	Greedy	Greedy
Selection	Smallest Files	Lowest Frequencies
Goal	Minimum Merge Cost	Minimum Encoding Cost
Data Structure	Min Heap	Min Heap

O. RGPV Exam Answers

2 Mark Answer

Optimal Merge Pattern is a Greedy algorithm used to merge multiple files with minimum total cost by repeatedly merging the two smallest files.

---

5 Mark Answer

Optimal Merge Pattern is a Greedy method used for minimizing file merge cost.

Steps:

1. Select two smallest files.

2. Merge them.
3. Add merge cost.
4. Insert merged file back.
5. Repeat until one file remains.

Applications:

- External Sorting
- Database Systems

---

7 Mark Answer

Explain Optimal Merge Pattern

Optimal Merge Pattern is a Greedy strategy used to merge files with minimum total cost.

Algorithm:

1. Select smallest two files.
2. Merge them.
3. Add merge cost.
4. Reinsert merged file.
5. Repeat.

Example:

Files:

```
10 20 30 40
```

Cost:

```
30 + 60 + 100
= 190
```

Complexity:

O(nlogn)O(n\log n)O(nlogn)

---

10 Mark Topper Answer

Optimal Merge Pattern

Definition

Optimal Merge Pattern is a Greedy algorithm that minimizes total file merging cost by repeatedly merging the smallest two files.

Greedy Choice

Choose smallest files first.

Algorithm

1. Insert all files.
2. Select two smallest files.
3. Merge them.
4. Add merge cost.
5. Insert merged file.
6. Repeat.

Example

Files:

```
10 20 30 40
```

Total Cost:

```
190
```

Complexity

O(nlogn)

Applications

- External Sorting
- Database Systems
- Data Compression

Conclusion

Optimal Merge Pattern provides minimum merging cost using the Greedy strategy.

P. One Page Revision Sheet

Optimal Merge Pattern

Goal

Minimum Merge Cost

Greedy Rule

Smallest Two Files First

Steps

Select Smallest Two

↓

Merge

↓

Add Cost

↓

Insert Again

↓

Repeat

Example

```
10 20 30 40
```

Cost:

```
30 + 60 + 100

= 190
```

Complexity

O(nlogn)

Data Structure

Min Heap

Priority Queue

Memory Trick

SSF

Small + Small First

Most Important Question

🔥 **Explain Optimal Merge Pattern with suitable example and complexity analysis.**

Huffman Coding

(RGPV Exam-Oriented Complete Notes | Easy Hinglish)

A. Introduction

Suppose tumhare paas message hai:

```
AAAAABBBCCDE
```

Yahan:

- A bahut baar aaya hai
- E sirf 1 baar aaya hai

Agar har character ko same number of bits dein, memory waste hogi.

Question:

Kya frequently used characters ko chhota code aur less frequent characters ko bada code de sakte hain?

✅ Yes

Isi idea par **Huffman Coding** based hai.

Real-Life Analogy

Socho:

Daily use words:

- Hi
- Yes
- No

Inke shortcut hote hain.

Rare words ke shortcut nahi hote.

Huffman bhi frequently used characters ko short code deta hai.

B. Definition

Exam Definition

Huffman Coding is a Greedy algorithm used for lossless data compression by assigning variable-length binary codes to characters based on their frequencies.

Easy Explanation

Huffman Coding:

Jiska frequency zyada ho usko chhota code do.

Jiska frequency kam ho usko bada code do.

Result:

✔ Less storage

✔ Faster transmission

Why Needed?

Without Compression:

```
More Memory
More Storage
More Transmission Time
```

With Huffman:

```
Less Memory
Less Storage
Less Transmission Time
```

C. Core Concept

Keyword 1: Frequency

Character kitni baar aaya.

Example:

Character	Frequency
A	5
B	3
C	2
D	1

Keyword 2: Huffman Tree

Special Binary Tree used for coding.

Keyword 3: Leaf Node

Contains actual characters.

Keyword 4: Internal Node

Contains sum of frequencies.

Greedy Choice Property

Always choose:

Two nodes with smallest frequencies

Optimal Substructure

Small optimal merges create final optimal Huffman tree.

Why Greedy Works?

Because:

Lowest frequency characters should get longest codes.

Highest frequency characters should get shortest codes.

Selecting smallest frequencies first ensures minimum total bits.

D. Working

General Steps

Step 1

Calculate frequencies.

Step 2

Insert frequencies into Min Heap.

Step 3

Select two minimum frequencies.

Step 4

Merge them.

Step 5

Insert merged node back.

Step 6

Repeat until only one node remains.

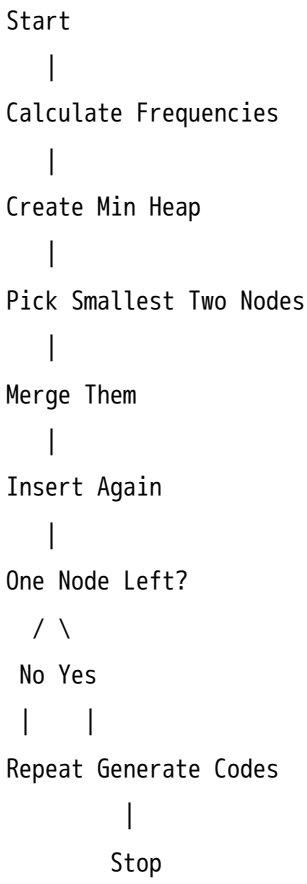
Step 7

Generate codes:

Left Edge = 0

Right Edge = 1

Flowchart



Pseudocode

```
Huffman(F)

Insert all frequencies in Min Heap

while more than one node exists

    x = smallest node

    y = second smallest node

    create node z

    z.freq = x.freq + y.freq

    insert z

Generate codes from tree
```

E. Dry Run (Most Important)

Example

Character	Frequency
A	5
B	9
C	12
D	13

Character	Frequency
E	16
F	45

Step 1

Pick smallest:

```
5 and 9
```

Merge:

```
14
```

Remaining:

```
12 13 14 16 45
```

Step 2

Pick:

```
12 and 13
```

Merge:

```
25
```

Remaining:

```
14 16 25 45
```

Step 3

Pick:

```
14 and 16
```

Merge:

```
30
```

Remaining:

```
25 30 45
```

Step 4

Pick:

```
25 and 30
```

Merge:

```
55
```

Remaining:

```
45 55
```

Step 5

Pick:

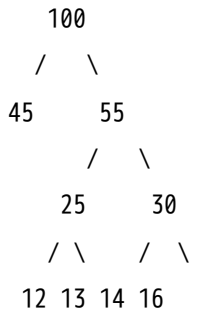
```
45 and 55
```

Merge:

```
100
```

Tree complete.

Huffman Tree



Assign Codes

Left = 0

Right = 1

Codes:

Character	Code
F	0
C	100
D	101
A	1100
B	1101
E	111

Observation

Highest frequency:

```
F = 45
```

Shortest code:

```
0
```

Exactly what we want.

F. Complexity Analysis

Using Min Heap:

Building Heap:

O(n)

Each Merge:

O(logn)

Total Merges:

```
n-1
```

Total Complexity:

O(nlogn)

Space Complexity

O(n)

G. Advantages

1

Efficient compression.

2

Lossless compression.

(No data loss)

3

Reduces storage.

4

Reduces transmission time.

5

Optimal prefix codes.

H. Disadvantages

1

Tree construction required.

2

Extra memory required.

3

Complex compared to fixed-length coding.

I. Applications

ZIP Compression

File compression.

JPEG

Image compression (partially).

MP3

Audio compression.

Data Transmission

Network communication.

Text Compression

Large text files.

J. Common Mistakes

❌ Selecting largest frequencies first.

✔ Select smallest frequencies first.

❌ Forgetting Min Heap.

✔ Mention Min Heap.

❌ Writing fixed length codes.

✔ Huffman uses variable-length codes.

K. Viva Questions

What is Huffman Coding?

A lossless compression technique.

Which technique is used?

Greedy Method.

Which data structure is used?

Min Heap.

Why is Huffman efficient?

Frequently used characters get shorter codes.

Complexity?

O(nlogn)

L. Exam Keywords

Write these words:

- Greedy Strategy
- Lossless Compression
- Variable Length Codes
- Huffman Tree
- Min Heap
- Frequency
- Prefix Code
- Binary Tree
- Compression
- Data Encoding

M. Memory Trick

Huffman Rule

Small + Small First

Same as Optimal Merge Pattern.

Coding Rule

```
Left = 0

Right = 1
```

Remember

Highest Frequency

↓

Shortest Code

N. Comparison Table

Huffman vs Fixed-Length Coding

Feature	Huffman	Fixed Length
Code Size	Variable	Fixed
Compression	Yes	No
Memory Usage	Less	More
Efficiency	High	Low

O. RGPV Exam Answers

2 Mark Answer

Huffman Coding is a Greedy algorithm used for lossless data compression. It assigns shorter codes to high-frequency characters and longer codes to low-frequency characters.

---

5 Mark Answer

Huffman Coding is a Greedy method used for data compression.

Steps:

1. Calculate frequencies.
2. Create Min Heap.
3. Select two minimum frequencies.
4. Merge them.
5. Repeat.
6. Generate codes.

Applications:

- ZIP
- JPEG
- MP3

---

7 Mark Answer

Explain Huffman Coding

Huffman Coding is a lossless compression algorithm based on Greedy strategy.

It assigns shorter codes to frequently occurring characters.

Algorithm:

1. Insert frequencies in Min Heap.
2. Pick smallest two frequencies.
3. Merge them.
4. Repeat.
5. Generate Huffman codes.

Complexity:

O(nlogn)

Applications:

- Data Compression
- ZIP
- JPEG

---

10 Mark Topper Answer

Huffman Coding

Definition

Huffman Coding is a Greedy algorithm used for lossless compression.

Greedy Choice

Always select two minimum frequencies.

Algorithm

1. Create Min Heap.
2. Pick two smallest nodes.
3. Merge them.
4. Insert merged node.
5. Repeat.
6. Generate codes.

Tree Construction

(Draw Huffman Tree)

Complexity

O(nlogn)

Advantages

- Efficient
- Lossless
- Minimum average code length

Applications

- ZIP
- JPEG
- MP3
- Data Transmission

Conclusion

Huffman Coding is one of the most widely used compression techniques based on Greedy Method.

P. One Page Revision Sheet

Huffman Coding

Goal:

Data Compression

Technique

Greedy Method

Data Structure

Min Heap

Greedy Rule

Smallest Two Frequencies First

Coding Rule

Left = 0

Right = 1

Complexity

O(nlogn)O(n\log n)O(nlogn)

Applications

- ZIP
- JPEG
- MP3

Most Important Points

Highest Frequency

↓

Shortest Code

Memory Trick

SSF

Small + Small First

Most Important Question

🔥 **Explain Huffman Coding with Huffman Tree construction and complexity analysis.**

Minimum Spanning Tree (MST)

(RGPV Exam-Oriented Complete Notes | Easy Hinglish)

A. Introduction

Suppose tumhe 5 cities ko roads se connect karna hai.

Goal:

✅ Sabhi cities connect ho jayein

✅ Minimum cost lage

Question:

Aisi road network kaise banaye jisme total cost sabse kam ho?

Is problem ka solution hai:

Minimum Spanning Tree (MST)

Real-Life Analogy

Imagine:

5 villages ko electricity connection dena hai.

Har village ke beech wire lagane ki cost alag hai.

Hume:

✔ Sab villages connect karne hain

✔ Minimum wire use karni hai

✔ Unnecessary loops avoid karne hain

Yahi MST ka concept hai.

B. Definition

Exam Definition

A Minimum Spanning Tree (MST) is a spanning tree of a connected weighted graph having minimum total edge weight.

Easy Explanation

MST ek aisa tree hai jo:

1. Sab vertices ko connect kare
2. Cycle na banaye
3. Minimum total cost rakhe

C. Core Concept

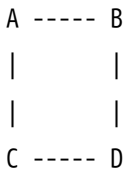
Before MST, kuch basic terms samjho.

---

1. Graph

Graph = Vertices + Edges

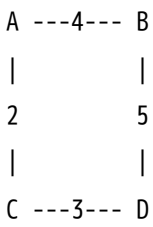
Example:



---

2. Weighted Graph

Har edge ka weight/cost hota hai.



---

3. Tree

Connected graph without cycle.

---

4. Spanning Tree

Graph ke sab vertices include hone chahiye.

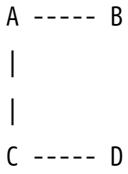
Rules:

- All vertices connected
- No cycle
- Exactly (V−1) edges

Example

Graph:

Spanning Tree:



All vertices connected.

No cycle.

---

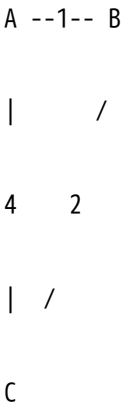
5. Minimum Spanning Tree

Agar multiple spanning trees possible ho:

Choose the one with minimum cost.

Example

Graph



Possible Spanning Tree 1

A-B = 1

B-C = 2

Cost:

1+2=3

Possible Spanning Tree 2

A-B = 1

A-C = 4

Cost:

1+4=5

Minimum Cost:

3

Therefore MST selected.

Properties of MST

Property 1

Contains all vertices.

Property 2

Contains exactly:

V−1V-1V−1

edges.

Property 3

No cycles.

Property 4

Minimum total edge weight.

Why MST is Important?

Many real-world optimization problems can be modeled using MST.

D. Working

There are two popular algorithms:

1. Prim's Algorithm
2. Kruskal's Algorithm

Both use Greedy Strategy.

Greedy Choice Property

Always choose:

Minimum cost edge

without violating MST conditions.

Optimal Substructure

Small minimum-cost connections combine to form the final MST.

General Procedure

Step 1

Start with graph.

Step 2

Choose minimum edge.

Step 3

Avoid cycles.

Step 4

Continue until:

```
V-1 edges selected
```

Step 5

Tree complete.

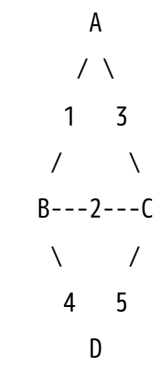
Flowchart

```
Start
  |
Input Graph
  |
```

```
Select Minimum Edge
  |
Forms Cycle?
 / \
Yes No
 |   |
Reject Add
 |   |
Repeat
  |
V-1 edges selected?
 / \
No Yes
 |   |
Repeat Stop
```

E. Dry Run Example

Graph:



Edges:

```
AB = 1
BC = 2
```

AC = 3

BD = 4

CD = 5

Step 1

Select smallest edge.

AB = 1

Cost = 1

Step 2

Select next smallest.

BC = 2

Cost = 3

Step 3

Select next smallest.

AC = 3

Creates cycle.

Reject.

Step 4

Select next edge.

```
BD = 4
```

Cost = 7

MST Obtained

```
A---1---B---2---C
        |
        4
        |
        D
```

Total Cost:

```
7
```

F. Complexity Analysis

MST itself is a concept.

Complexity depends on algorithm.

Prim's Algorithm

Using Min Heap:

O(ElogV)

Kruskal's Algorithm

Using Sorting:

O(ElogE)

Space Complexity

Usually:

O(V+E)

G. Advantages

1

Minimum cost network.

2

Efficient resource utilization.

3

Simple optimization model.

4

Useful in networking.

H. Disadvantages

1

Applicable only to connected graphs.

2

Complex implementation for large graphs.

3

Needs weighted graph.

I. Applications

Computer Networks

Minimum cable length.

Telephone Networks

Minimum wiring.

Electricity Distribution

Power lines.

Road Construction

Minimum road cost.

Water Pipeline Systems

Minimum pipe length.

J. Common Mistakes

❌ MST = Graph

✔ MST is a tree derived from graph.

❌ MST may contain cycles

✔ MST never contains cycles.

❌ MST can have V edges

✔ MST always has:

```
V-1 edges
```

K. Viva Questions

What is MST?

Minimum Spanning Tree.

What is Spanning Tree?

Tree connecting all vertices.

How many edges in MST?

V−1

Does MST contain cycles?

No.

Which algorithms find MST?

- Prim's Algorithm
- Kruskal's Algorithm

L. Exam Keywords

Write these keywords:

- Connected Graph
- Weighted Graph
- Minimum Cost
- Spanning Tree
- No Cycles
- V−1 Edges
- Greedy Method
- Prim's Algorithm
- Kruskal's Algorithm

- Optimization

M. Memory Trick

MST Rule

Connect All + Minimum Cost + No Cycle

Mnemonic:

CMN

Connect

Minimum Cost

No Cycle

N. Comparison Table

Spanning Tree vs MST

Feature	Spanning Tree	MST
Connects all vertices	Yes	Yes
No cycle	Yes	Yes
Minimum cost	Not necessary	Yes
Number of edges	V−1	V−1

O. RGPV Exam Answers

2 Mark Answer

A Minimum Spanning Tree (MST) is a spanning tree of a connected weighted graph having minimum total edge weight.

---

5 Mark Answer

MST is a spanning tree that connects all vertices with minimum cost and without cycles.

Properties:

1. Connected graph
2. No cycles
3. V−1 edges
4. Minimum total weight

Applications:

- Networking
- Road construction
- Power distribution

---

7 Mark Answer

Explain MST

MST stands for Minimum Spanning Tree.

It is a spanning tree of a weighted graph with minimum total edge weight.

Properties:

- Contains all vertices
- Contains V−1 edges
- No cycles
- Minimum cost

Algorithms:

- Prim's Algorithm
- Kruskal's Algorithm

Applications:

- Networks
- Electricity distribution
- Road systems

---

10 Mark Topper Answer

Minimum Spanning Tree

Definition

A Minimum Spanning Tree is a spanning tree of a connected weighted graph that connects all vertices with minimum total edge weight.

Properties

1. Connected graph
2. $V-1$ edges
3. No cycles
4. Minimum total cost

Greedy Principle

Choose minimum weight edges while avoiding cycles.

Algorithms

- Prim's Algorithm
- Kruskal's Algorithm

Applications

- Computer Networks
- Telephone Networks

- Transportation Systems
- Power Distribution

Conclusion

MST is an important optimization concept used to minimize connection costs in weighted graphs.

P. One Page Revision Sheet

MST

Minimum Spanning Tree

Rules

✔ Connect All Vertices

✔ No Cycle

✔ Minimum Cost

✔ V−1 Edges

Algorithms

1. Prim's Algorithm
2. Kruskal's Algorithm

Formula

Edges=V−1

Applications

- Networks
- Roads
- Power Lines
- Pipelines

Memory Trick

CMN

Connect

Minimum Cost

No Cycle

Most Important Question

🔥 **Explain Minimum Spanning Tree (MST) and its properties with suitable example.**

Prim's Algorithm

(RGPV Exam-Oriented Complete Notes | Easy Hinglish)

A. Introduction

Prim's Algorithm ka use **Minimum Spanning Tree (MST)** banane ke liye kiya jata hai.

MST ka goal:

✔ Sab vertices connect ho jayein

✔ Cycle na bane

✔ Total cost minimum ho

Real-Life Analogy

Maan lo tumhe 5 cities ko roads se connect karna hai.

Har road ki cost alag hai.

Tumhara goal:

Sab cities connect ho jayein aur total road cost minimum ho.

Prim's Algorithm exactly yehi karta hai.

B. Definition

Exam Definition

Prim's Algorithm is a Greedy algorithm used to find the Minimum Spanning Tree of a connected weighted graph by repeatedly selecting the minimum weight edge connecting a visited vertex to an unvisited vertex.

Easy Explanation

Prim's Algorithm:

👉 Kisi ek vertex se start karo.

👉 Har step par minimum cost edge choose karo.

👉 Ek-ek karke MST grow karo.

C. Core Concept

Greedy Choice Property

Har step par:

Minimum weight edge choose karo.

Optimal Substructure

Har minimum edge future MST ka part ban sakti hai.

Key Idea

Prim's Algorithm:

```
Vertex se start karo

↓

Minimum edge choose karo

↓

New vertex add karo

↓

Repeat
```

D. Working

Steps

Step 1

Kisi bhi vertex se start karo.

Step 2

Visited aur Unvisited vertices maintain karo.

Step 3

Visited se Unvisited tak ki minimum edge choose karo.

Step 4

Edge MST mein add karo.

Step 5

Naya vertex visited set mein add karo.

Step 6

Repeat until:

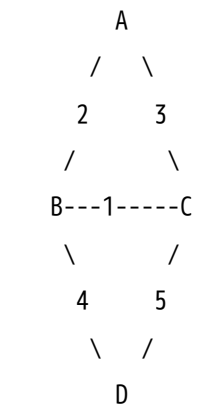
```
V - 1 edges selected
```

Flowchart

```
Start
  |
Choose Start Vertex
  |
Find Minimum Edge
  |
Add Edge to MST
  |
Add New Vertex
  |
All Vertices Visited?
 / \
No Yes
 |   |
Repeat Stop
```

E. Dry Run (Most Important)

Example Graph



Edge List

```
AB = 2

AC = 3

BC = 1

BD = 4

CD = 5
```

Step 1

Start from:

```
A
```

Visited:

```
{A}
```

Step 2

Possible edges:

```
AB = 2
```

```
AC = 3
```

Choose minimum:

```
AB = 2
```

MST Cost:

```
2
```

Visited:

```
{A,B}
```

Step 3

Possible edges:

```
AC = 3
```

```
BC = 1
```

```
BD = 4
```

Choose minimum:

```
BC = 1
```

MST Cost:

```
2 + 1 = 3
```

Visited:

```
{A,B,C}
```

Step 4

Possible edges:

```
BD = 4

CD = 5
```

Choose minimum:

```
BD = 4
```

MST Cost:

```
3 + 4 = 7
```

Visited:

```
{A,B,C,D}
```

Final MST

```
A---2---B---1---C
        |
        4
        |
        D
```

Total Cost

```
2 + 1 + 4 = 7
```

F. Algorithm

Pseudocode

```
Prim(G)

Choose starting vertex

while MST not complete

    Select minimum edge

    connecting visited vertex

    to unvisited vertex

    Add edge to MST

    Mark vertex visited

return MST
```

G. Complexity Analysis

Using Adjacency Matrix

Complexity:

O(V^2)

Using Min Heap + Adjacency List

Complexity:

O(ElogV)

Best Case

O(ElogV)

Average Case

O(ElogV)

Worst Case

O(ElogV)

Space Complexity

O(V)

Why O(E log V)?

Each edge is processed.

Priority Queue operations:

```
Insert
```

```
Delete-Min
```

take:

```
O(log V)
```

Therefore:

```
O(E log V)
```

H. Advantages

1

Produces MST.

2

Greedy and efficient.

3

Works well for dense graphs.

4

Simple implementation.

I. Disadvantages

1

Works only on connected graphs.

2

Priority Queue required for efficiency.

3

Not suitable for disconnected graphs.

J. Applications

Computer Networks

Minimum cable cost.

Telephone Networks

Minimum wiring.

Road Construction

Minimum road length.

Water Supply Systems

Minimum pipe network.

Electrical Networks

Minimum transmission cost.

K. Common Mistakes

❌ Choosing globally smallest edge.

✔ Prim chooses smallest edge connected to visited set.

❌ Forgetting visited/unvisited concept.

✔ Always mention visited and unvisited vertices.

❌ Creating cycle.

✔ MST never contains cycles.

L. Viva Questions

What is Prim's Algorithm?

Greedy algorithm for MST.

Which technique is used?

Greedy Method.

What does it generate?

Minimum Spanning Tree.

Does Prim create cycles?

No.

Complexity?

Using Heap:

O(ElogV)

M. Exam Keywords

Write these words:

- Minimum Spanning Tree
- Greedy Strategy
- Connected Graph
- Minimum Edge
- Visited Vertex
- Unvisited Vertex
- Priority Queue
- No Cycles
- MST Cost
- Optimization

N. Memory Trick

Prim Rule

"Grow the Tree"

Start from one vertex.

Add nearest vertex.

Repeat.

Mnemonic:

```
S M A R T
```

Start

Minimum Edge

Add Vertex

Repeat

Tree Complete

O. Comparison Table

Prim vs Kruskal

Feature	Prim	Kruskal
Starts From	Vertex	Edge
Builds	Single Tree	Forest
Cycle Check	Not explicit	Required
Suitable For	Dense Graph	Sparse Graph
Technique	Greedy	Greedy

P. RGPV Exam Answers

2 Mark Answer

Prim's Algorithm is a Greedy algorithm used to find the Minimum Spanning Tree of a connected weighted graph by repeatedly selecting the minimum weight edge.

---

5 Mark Answer

Prim's Algorithm constructs MST by starting from any vertex and repeatedly selecting the minimum cost edge connecting a visited vertex to an unvisited vertex.

Steps:

1. Start from a vertex.
2. Select minimum edge.
3. Add new vertex.
4. Repeat.

Applications:

- Computer Networks
- Road Systems

---

7 Mark Answer

Explain Prim's Algorithm

Prim's Algorithm is a Greedy method for finding MST.

Working:

1. Select start vertex.
2. Choose minimum edge.
3. Add connected vertex.
4. Repeat until MST complete.

Example:

Graph:

```
AB = 2
BC = 1
BD = 4
```

MST Cost:

```
7
```

Complexity:

O(ElogV)

---

10 Mark Topper Answer

Prim's Algorithm

Definition

Prim's Algorithm is a Greedy algorithm that constructs a Minimum Spanning Tree by repeatedly selecting the minimum cost edge connecting the current tree to a new vertex.

Algorithm

1. Choose start vertex.
2. Select minimum edge.
3. Add new vertex.
4. Repeat.

Example

(Draw graph and MST)

Complexity

Adjacency Matrix:

O(V^2)

Using Heap:

O(ElogV)

Applications

- Networks
- Roads
- Power Distribution

Conclusion

Prim's Algorithm efficiently constructs a Minimum Spanning Tree using the Greedy approach.

One Page Revision Sheet

Prim's Algorithm

Goal:

Minimum Spanning Tree

Technique

Greedy Method

Rule

Choose Minimum Edge

Visited → Unvisited

Steps

Start Vertex

↓

Minimum Edge

↓

Add Vertex

↓

Repeat

Complexity

Matrix:

O(V^2)

Heap:

O(ElogV)

Applications

- Networks
- Roads
- Pipelines
- Electricity

Memory Trick

SMART

Start

Minimum Edge

Add Vertex

Repeat

Tree Complete

Most Important Question

🔥 **Explain Prim's Algorithm with suitable example and complexity analysis.**

Fractional Knapsack Problem

(RGPV Exam-Oriented Complete Notes | Easy Hinglish)

A. Introduction

Maan lo tumhare paas ek bag (Knapsack) hai.

Bag Capacity = **50 kg**

Aur kuch items hain:

Item	Profit	Weight
A	60	10
B	100	20
C	120	30

Question:

Bag mein kaunse items rakhein taki **maximum profit** mile?

Agar item ko todkar (fraction mein) bhi le sakte hain, to ye problem **Fractional Knapsack Problem** kehlati hai.

Real-Life Analogy

Suppose tum fruit shop se dry fruits kharid rahe ho.

Tum 1 kg badam ke bajay 0.5 kg bhi le sakte ho.

Yani item ko fractional part mein liya ja sakta hai.

Isi wajah se Greedy Method yahan perfectly work karta hai.

B. Definition

Exam Definition

Fractional Knapsack is a Greedy optimization problem in which items can be divided into fractions, and the objective is to maximize total profit within the given knapsack capacity.

Easy Explanation

Rule:

👉 Item ko pura lena zaroori nahi.

👉 Jitna part chahiye utna le sakte ho.

👉 Highest profit/weight ratio wale item ko pehle lo.

C. Core Concept

Keyword 1: Profit

Item bechne ya rakhne se milne wala benefit.

Keyword 2: Weight

Item ka weight.

Keyword 3: Capacity

Knapsack kitna weight carry kar sakta hai.

Keyword 4: Profit/Weight Ratio

Most Important Formula:

Profit/Weight

Greedy Choice Property

Always choose item with:

Maximum Profit/Weight Ratio

Optimal Substructure

Best item selections milkar maximum overall profit dete hain.

Why Greedy Works?

Because:

Profit per kg sabse zyada dene wale item ko pehle lena logical hai.

Isse maximum benefit milta hai.

D. Working

General Steps

Step 1

Calculate Profit/Weight ratio.

Step 2

Sort items in decreasing order of ratio.

Step 3

Pick highest ratio item.

Step 4

If item fits completely:

Take full item.

Step 5

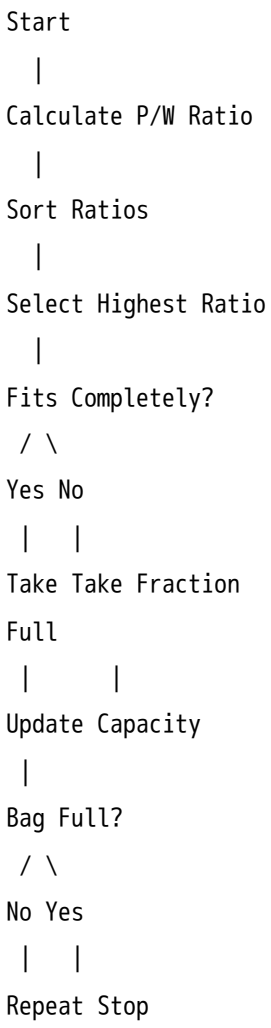
If item does not fit:

Take required fraction.

Step 6

Stop when bag becomes full.

Flowchart



E. Dry Run (Most Important)

Example

Knapsack Capacity:

```
50 kg
```

Items:

Item	Profit	Weight
A	60	10
B	100	20
C	120	30

Step 1

Calculate Ratio

Item	P/W
A	6
B	5
C	4

Step 2

Sort

```
A → B → C
```

Step 3

Take A

Weight used:

```
10
```

Profit:

```
60
```

Remaining Capacity:

```
40
```

Step 4

Take B

Weight used:

```
20
```

Profit:

```
100
```

Remaining Capacity:

```
20
```

Total Profit:

```
160
```

Step 5

Item C weight = 30

Only 20 kg space left.

Take fraction:

```
20/30
```

of item C.

Profit from fraction:

```
120 × (20/30)
```

```
= 80
```

Final Profit

```
60 + 100 + 80

= 240
```

Final Answer

Maximum Profit:

```
240
```

F. Algorithm

Pseudocode

```
FractionalKnapsack(items,W)

Sort items by Profit/Weight ratio

profit = 0

for each item
```

```
   if item fits
      take full item
      profit += item profit
   else
      take fraction
      profit += fractional profit
      break
return profit
```

G. Complexity Analysis

Sorting Items

Complexity:

O(nlogn)

Selection

Complexity:

O(n)

Total Complexity

O(nlogn)

Best Case

O(nlogn)

Average Case

O(nlogn)

Worst Case

O(nlogn)

Space Complexity

O(1)

(Extra memory very small)

Why O(n log n)?

Sorting dominates execution time.

Therefore:

O(nlogn)

H. Advantages

1

Simple Greedy solution.

2

Produces optimal solution.

3

Fast execution.

4

Easy implementation.

I. Disadvantages

1

Works only when items are divisible.

2

Cannot solve 0/1 Knapsack.

3

Not suitable when items are indivisible.

J. Applications

Cargo Loading

Ship loading.

Investment Problems

Capital allocation.

Resource Allocation

Limited resources.

Budget Optimization

Maximum benefit selection.

K. Common Mistakes

❌ Sorting by profit.

✔ Sort by profit/weight ratio.

❌ Taking random items.

✔ Always choose highest ratio.

❌ Confusing Fractional with 0/1 Knapsack.

✔ Fractional allows division.

✔ 0/1 does not.

L. Viva Questions

What is Fractional Knapsack?

A Greedy optimization problem.

Which property is used?

Profit/Weight Ratio.

Can item be divided?

Yes.

Technique used?

Greedy Method.

Complexity?

O(nlogn)

M. Memory Trick

Knapsack Rule

Highest Ratio First

Mnemonic:

```
P/W
```

Profit per Weight

Remember:

More Profit

Less Weight

↓

Choose First

N. Comparison Table

Fractional vs 0/1 Knapsack

Feature	Fractional	0/1 Knapsack
Item Division	Allowed	Not Allowed
Technique	Greedy	Dynamic Programming
Optimal by Greedy	Yes	No
Complexity	O(n log n)	O(nW)
Item Selection	Full/Fraction	Full Only

O. RGPV Exam Answers

2 Mark Answer

Fractional Knapsack is a Greedy optimization problem where items can be divided into fractions and selected to maximize profit within a given capacity.

---

5 Mark Answer

Fractional Knapsack uses the Greedy approach by selecting items based on maximum Profit/Weight ratio.

Steps:

1. Calculate P/W ratio.
2. Sort items.
3. Select highest ratio items.
4. Take fraction if required.

Complexity:

O(nlogn)

---

7 Mark Answer

Explain Fractional Knapsack

Fractional Knapsack is a Greedy optimization problem.

Algorithm:

1. Calculate Profit/Weight ratio.
2. Sort descending.
3. Select highest ratio item.
4. Take fraction if needed.

Example:

Capacity = 50

Profit obtained:

Complexity:

O(nlogn)

---

10 Mark Topper Answer

Fractional Knapsack Problem

Definition

Fractional Knapsack is a Greedy optimization technique where items can be divided and selected according to Profit/Weight ratio.

Greedy Choice

Choose highest Profit/Weight ratio.

Algorithm

1. Compute ratio.
2. Sort.
3. Select items.
4. Take fraction if necessary.

Example

(Draw item table and ratio table)

Maximum Profit:

```
240
```

Complexity

O(nlogn)O(n\log n)O(nlogn)

Applications

- Cargo loading
- Investment planning
- Resource allocation

Conclusion

Fractional Knapsack provides optimal profit using Greedy strategy.

P. One Page Revision Sheet

Fractional Knapsack

Goal:

Maximum Profit

Technique

Greedy Method

Formula

Profit/Weight

Rule

Highest Ratio First

Steps

Calculate Ratio

↓

Sort

↓

Select

↓

Take Fraction

↓

Stop

Complexity

O(nlogn)O(n\log n)O(nlogn)

Difference

Fractional → Division Allowed

0/1 → Division Not Allowed

Memory Trick

P/W First

Profit per Weight First

Most Important Question

🔥 **Explain Fractional Knapsack Problem with suitable example and complexity analysis.**

Job Sequencing with Deadlines

(RGPV Exam-Oriented Complete Notes | Easy Hinglish)

A. Introduction

Maan lo tum ek freelancer ho.

Tumhare paas 4 projects hain.

Har project:

- Profit deta hai
- Deadline hoti hai

Tum ek time par sirf **1 job** kar sakte ho.

Question:

👉 Kaunsi jobs select karni chahiye taki **maximum profit** mile?

Is problem ko solve karta hai:

Job Sequencing with Deadlines

Real-Life Analogy

Suppose:

Job	Profit	Deadline
Website	₹100	Day 2
Poster	₹50	Day 1
Logo	₹70	Day 2

Tum sab jobs nahi kar sakte.

Goal:

Maximum profit earn karna.

B. Definition

Exam Definition

Job Sequencing with Deadlines is a Greedy optimization problem in which jobs having deadlines and profits are scheduled to maximize total profit.

Easy Explanation

Har job:

- 1 unit time leti hai.
- Ek deadline hoti hai.
- Ek profit hota hai.

Rule:

👉 Highest profit jobs ko pehle schedule karo.

Assumptions

1. Har job = 1 unit time.
2. Ek slot mein sirf ek job.
3. Deadline miss hui to profit = 0.

C. Core Concept

Keyword 1: Job

Work to be completed.

Keyword 2: Deadline

Last time slot before which job must finish.

Keyword 3: Profit

Job complete karne par milne wala reward.

Greedy Choice Property

Always choose:

Highest Profit Job First

Optimal Substructure

Best scheduled jobs milkar maximum total profit deti hain.

Why Greedy Works?

Agar high-profit jobs pehle schedule karenge to overall profit maximize hoga.

D. Working

General Steps

Step 1

Sort jobs in decreasing order of profit.

Step 2

Find maximum deadline.

Step 3

Create time slots.

Step 4

Take highest profit job.

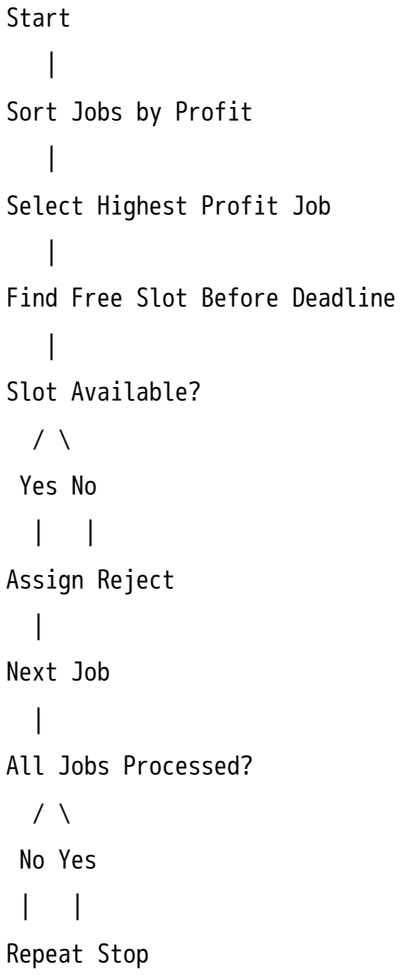
Step 5

Place it in latest available slot before its deadline.

Step 6

Repeat for remaining jobs.

Flowchart



E. Dry Run (Most Important)

Example

Job	Deadline	Profit
J1	2	100
J2	1	19
J3	2	27
J4	1	25
J5	3	15

Step 1

Sort by Profit

Job	Deadline	Profit
J1	2	100
J3	2	27
J4	1	25
J2	1	19
J5	3	15

Step 2

Maximum Deadline

```
3
```

Create Slots

```
Slot1 Slot2 Slot3
```

Initially:

```
_ _ _
```

Step 3

Take J1

Deadline = 2

Place at Slot2

```
_ J1 _
```

Profit:

```
100
```

Step 4

Take J3

Deadline = 2

Slot2 occupied.

Try Slot1.

Place J3.

```
J3 J1 _
```

Profit:

Step 5

Take J4

Deadline = 1

Slot1 occupied.

Reject.

Step 6

Take J2

Deadline = 1

Slot1 occupied.

Reject.

Step 7

Take J5

Deadline = 3

Slot3 free.

Place J5.

```
J3 J1 J5
```

Profit:

```
127 + 15

= 142
```

Final Schedule

Slot	Job
1	J3
2	J1
3	J5

Maximum Profit

```
142
```

F. Algorithm

Pseudocode

```
JobSequencing(Jobs)

Sort jobs by profit

for each job

   find latest free slot

   before deadline

   if slot exists

      assign job

return schedule
```

G. Complexity Analysis

Sorting Jobs

Complexity:

O(nlogn)

Slot Searching

Worst Case:

O(n^2)

Total Complexity

Simple implementation:

O(n^2)

Best Case

O(nlogn)

Average Case

O(n^2)

Worst Case

O(n^2)

Space Complexity

O(n)

Why $O(n^2)$?

For every job:

May need to search multiple slots.

Therefore:

$n \times n$

↓

O(n^2)

H. Advantages

1

Maximizes profit.

2

Simple Greedy solution.

3

Efficient scheduling.

4

Easy implementation.

I. Disadvantages

1

Works only under given assumptions.

2

All jobs assumed equal duration.

3

Not suitable for complex scheduling.

J. Applications

CPU Scheduling

Process selection.

Project Management

Task scheduling.

Manufacturing Systems

Machine scheduling.

Resource Allocation

Optimal resource usage.

K. Common Mistakes

❌ Sorting by deadline.

✔ Sort by profit.

❌ Assigning earliest slot.

✔ Assign latest possible slot before deadline.

❌ Ignoring deadline.

✔ Deadline must be satisfied.

L. Viva Questions

What is Job Sequencing?

A Greedy scheduling problem.

Objective?

Maximum Profit.

Which technique is used?

Greedy Method.

How are jobs sorted?

By decreasing profit.

Complexity?

O(n^2)

M. Exam Keywords

Write these words:

- Greedy Strategy
- Deadline

- Profit
- Time Slot
- Scheduling
- Maximum Profit
- Job Assignment
- Optimization
- Latest Free Slot
- Resource Allocation

N. Memory Trick

Rule

Profit First

Not Deadline First.

Mnemonic:

```
SPD
```

Sort by Profit

↓

Place Job

↓

Deadline Check

O. Comparison Table

Job Sequencing vs Fractional Knapsack

Feature	Job Sequencing	Fractional Knapsack
Goal	Max Profit	Max Profit
Constraint	Deadline	Capacity
Sorting	Profit	Profit/Weight
Technique	Greedy	Greedy
Fraction Allowed	No	Yes

P. RGPV Exam Answers

2 Mark Answer

Job Sequencing with Deadlines is a Greedy optimization problem used to schedule jobs having deadlines and profits in order to maximize total profit.

---

5 Mark Answer

Job Sequencing schedules jobs in available time slots before their deadlines.

Steps:

1. Sort jobs by profit.
2. Select highest profit job.
3. Assign latest free slot before deadline.
4. Repeat.

Objective:

Maximum Profit.

---

7 Mark Answer

Explain Job Sequencing with Deadlines

Job Sequencing is a Greedy scheduling problem.

Algorithm:

1. Sort jobs by decreasing profit.
2. Create slots.
3. Assign each job to latest available slot.
4. Reject jobs if no slot available.

Example:

Final Schedule:

```
J3 J1 J5
```

Profit:

```
142
```

Complexity:

O(n^2)

---

10 Mark Topper Answer

Job Sequencing with Deadlines

Definition

Job Sequencing with Deadlines is a Greedy optimization problem used to maximize total profit while satisfying deadlines.

Greedy Choice

Choose highest profit job first.

Algorithm

1. Sort jobs by profit.
2. Create slots.
3. Assign jobs to latest free slot.
4. Reject conflicting jobs.

Example

(Write job table and scheduling process)

Complexity

Sorting:

O(nlogn)

Overall:

O(n^2)

Applications

- CPU Scheduling
- Project Management
- Resource Allocation

Conclusion

Job Sequencing efficiently maximizes profit using Greedy scheduling.

One Page Revision Sheet

Job Sequencing

Goal:

Maximum Profit

Technique

Greedy Method

Rule

Highest Profit First

Steps

Sort by Profit

↓

Find Latest Free Slot

↓

Assign Job

↓

Repeat

Complexity

Overall:

O(n^2)

Memory Trick

SPD

Sort Profit

Place Job

Deadline Check

Most Important Formula

Latest Free Slot Before Deadline

Most Important Question

🔥 **Explain Job Sequencing with Deadlines using Greedy Method with suitable example and complexity analysis.**

This is one of the most frequently asked 7–10 mark questions in ADA Unit-2.

Dijkstra's Single Source Shortest Path Algorithm

CS402 ADA – Unit 2 (Greedy Method)

Dijkstra's Single Source Shortest Path Algorithm (SSSP)

(RGPV Exam-Oriented Complete Notes | Easy Hinglish)

A. Introduction

Suppose tum **Bhopal** mein ho aur tumhe pata karna hai:

👉 Bhopal se Gwalior tak minimum distance kya hai?

👉 Bhopal se Indore tak shortest route kya hai?

👉 Bhopal se Jabalpur tak minimum cost kya hai?

Agar ek source vertex se graph ke sabhi vertices tak shortest path nikalna ho, to hum use karte hain:

Dijkstra's Algorithm

Real-Life Analogy

Google Maps kya karta hai?

Source:

```
Your Location
```

Destination:

```
Any City
```

Google shortest route batata hai.

Dijkstra bhi exactly yahi kaam karta hai.

B. Definition

Exam Definition

Dijkstra's Algorithm is a Greedy algorithm used to find the shortest path from a single source vertex to all other vertices in a weighted graph with non-negative edge weights.

Easy Explanation

Dijkstra ka goal:

👉 Ek source node choose karo.

👉 Har node tak minimum distance find karo.

Important Condition

Dijkstra only works when:

✔ Edge weights are positive (or zero)

❌ Negative weights allowed nahi hote.

C. Core Concept

Keyword 1: Source Vertex

Starting vertex.

Example:

A

Keyword 2: Destination Vertex

Jis vertex tak distance find karna hai.

Keyword 3: Path

Vertices ka sequence.

Keyword 4: Distance

Total edge weights ka sum.

Greedy Choice Property

Always choose:

Unvisited vertex having minimum distance

Optimal Substructure

Shortest paths milkar final shortest path tree banate hain.

Why Greedy Works?

Agar kisi vertex tak minimum distance mil gaya hai,

to future mein usse better distance nahi mil sakta.

Isliye us vertex ko permanently select kar sakte hain.

D. Working

Steps

Step 1

Source vertex choose karo.

Step 2

Source distance = 0

Baaki sab distances = ∞

Step 3

Minimum distance wala unvisited vertex select karo.

Step 4

Uske adjacent vertices update karo.

(Relaxation)

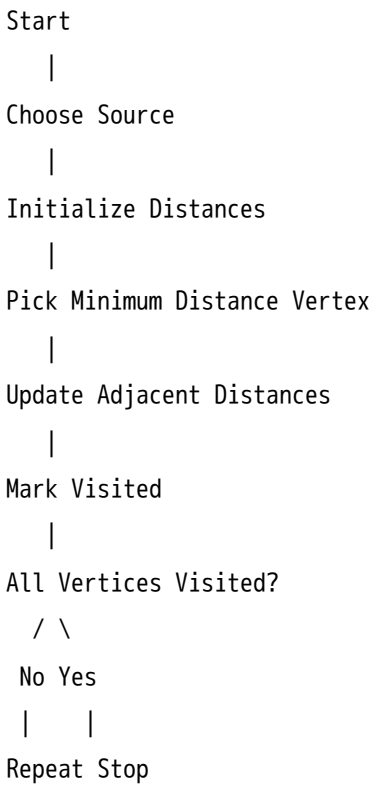
Step 5

Vertex ko visited mark karo.

Step 6

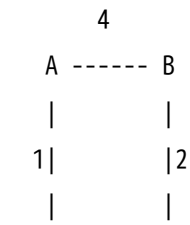
Repeat until all vertices visited.

Flowchart



E. Dry Run (Most Important)

Example Graph



```
  C ------ D
      5
```

Extra Edge:

```
B ----1---- C
```

Edge List

Edge	Weight
A-B	4
A-C	1
B-C	1
B-D	2
C-D	5

Source = A

Initial Table

Vertex	Distance
A	0
B	∞
C	∞
D	∞

Iteration 1

Choose:

```
A
```

Update neighbors:

```
B = 4
```

```
C = 1
```

Table:

Vertex	Distance
A	0
B	4
C	1
D	∞

Iteration 2

Minimum distance:

```
C = 1
```

Choose C.

Update:

```
B = min(4,1+1)
```

```
= 2
```

```
D = min(∞,1+5)
```

```
= 6
```

Table:

Vertex	Distance
A	0
B	2
C	1
D	6

Iteration 3

Choose:

```
B = 2
```

Update:

```
D = min(6,2+2)
```

```
= 4
```

Table:

Vertex	Distance
A	0
B	2
C	1
D	4

Iteration 4

Choose:

```
D = 4
```

Done.

Final Shortest Distances

Vertex	Distance
A	0
B	2
C	1
D	4

Shortest Path Tree

```
A
|
1
|
C
|
1
|
B
|
2
|
D
```

F. Algorithm

Pseudocode

```
Dijkstra(G,Source)

dist[source] = 0

all others = infinity

while vertices remain

   select minimum distance vertex

   mark visited

   update adjacent vertices

return distances
```

Relaxation Formula

Most Important Formula:

dist[v]=min(dist[v],dist[u]+w(u,v))

Where:

- dist[v] = current distance
- dist[u] = source to u distance
- w(u,v) = edge weight

G. Complexity Analysis

Using Adjacency Matrix

Complexity:

O(V^2)

Using Min Heap

Complexity:

O(ElogV)

Best Case

O(ElogV)

Average Case

O(ElogV)

Worst Case

O(ElogV)

Space Complexity

O(V)

H. Advantages

1

Finds shortest path efficiently.

2

Greedy and simple.

3

Used in networking.

4

Works well for positive weights.

I. Disadvantages

1

Does not support negative edge weights.

2

Not suitable for all graph types.

3

Bellman-Ford needed for negative weights.

J. Applications

GPS Navigation

Google Maps.

Computer Networks

Routing.

Airline Route Planning

Minimum travel cost.

Telecommunications

Shortest transmission path.

Logistics

Delivery route optimization.

K. Common Mistakes

❌ Using negative weights.

✔ Dijkstra works only for non-negative weights.

❌ Forgetting relaxation.

✔ Always update distances.

❌ Selecting random vertex.

✔ Select minimum distance vertex.

L. Viva Questions

What is Dijkstra's Algorithm?

Greedy shortest path algorithm.

What problem does it solve?

Single Source Shortest Path.

Does it allow negative weights?

No.

Which technique is used?

Greedy Method.

Complexity?

Heap version:

O(ElogV)

M. Exam Keywords

Write these words:

- Single Source Shortest Path
- Greedy Strategy
- Relaxation
- Minimum Distance
- Weighted Graph
- Non-Negative Weights
- Priority Queue
- Shortest Path Tree
- Optimization
- Routing

N. Memory Trick

Dijkstra Rule

Nearest Node First

Mnemonic:

S M U V

Source

Minimum Distance

Update Neighbors

Visit Vertex

O. Comparison Table

Dijkstra vs MST

Feature	Dijkstra	MST
Goal	Shortest Path	Minimum Cost Tree
Output	Distances	Tree
Technique	Greedy	Greedy
Algorithms	Dijkstra	Prim/Kruskal
Source Vertex	Required	Not Required

P. RGPV Exam Answers

2 Mark Answer

Dijkstra's Algorithm is a Greedy algorithm used to find the shortest path from a single source vertex to all other vertices in a weighted graph with non-negative edge weights.

---

5 Mark Answer

Dijkstra's Algorithm finds shortest paths from a source vertex.

Steps:

1. Initialize distances.
2. Select minimum distance vertex.
3. Update adjacent vertices.
4. Repeat.

Applications:

- Routing
- GPS Navigation

---

7 Mark Answer

Explain Dijkstra's Algorithm

Dijkstra's Algorithm is a Greedy method for solving the Single Source Shortest Path problem.

Algorithm:

1. Choose source vertex.
2. Initialize distances.
3. Select minimum distance vertex.
4. Update neighbors.
5. Repeat.

Complexity:

Heap version:

O(ElogV)

Applications:

- Computer Networks
- GPS Systems

---

10 Mark Topper Answer

Dijkstra's Single Source Shortest Path Algorithm

Definition

Dijkstra's Algorithm is a Greedy algorithm used to find shortest paths from a source vertex to all other vertices in a weighted graph.

Greedy Choice

Choose the unvisited vertex having minimum distance.

Relaxation Formula

dist[v]=min(dist[v],dist[u]+w(u,v))

Algorithm

1. Initialize distances.
2. Select minimum distance vertex.
3. Update neighbors.
4. Repeat until all vertices visited.

Complexity

Matrix:

O(V^2)

Heap:

O(ElogV)O(E\log V)O(ElogV)

Applications

- GPS Navigation
- Routing
- Network Optimization

Conclusion

Dijkstra's Algorithm efficiently finds shortest paths in graphs having non-negative edge weights.

One Page Revision Sheet

Dijkstra Algorithm

Goal:

Shortest Path

Technique

Greedy Method

Rule

Nearest Vertex First

Formula

dist[v]=min(dist[v],dist[u]+w(u,v))

Steps

Initialize

↓

Choose Minimum Distance Vertex

↓

Update Neighbors

↓

Repeat

Complexity

Matrix:

O(V^2)

Heap:

O(ElogV)

Limitation

❌ Negative Weights Not Allowed

Memory Trick

SMUV

Source

Minimum Distance

Update

Visit

Most Important Question

🔥 Explain Dijkstra's Single Source Shortest Path Algorithm with suitable example and complexity