studocu

DAA Final

Design and analysis of algorithm (Lovely Professional University)

Scan to open on Studocu

Studocu is not sponsored or endorsed by any college or university
Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Introduction

→ Algorithm
- → finite sequence of instructions
- → No ambiguity (clear meaning)
- → I/p : zero or more
- → O/p : Atleast one
- → Termination after finite no. of steps (for all cases)
- → solving a problem
- → Performing computation
- → Terminate in finite time
- → Effectivenes (✓)
- → Described Precisely

Example: Making of coffee

1. Add milk in vessel (milk → liquid) (vessel → kettle)
2. Add coffee powder
3. Add sugar
4. Mix it
5. serve and drink

(Not Described precisely)

(No mention of heat / put ON gas / boil)

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Algorithm Vs Program

Algorithm	Program
① Design	① Implementation
② PL (x)	② PL (✓)
③ Natural / Normal Language (SR x)	③ follow SR ✓
④ Logical	④ Expression
⑤ Analyze (T&S)	⑤ Test (Errors)
⑥ Independent of (H/w & S/w)	⑥ Dependent of (H/w & S/w)

⇒ Problem solving

- → Problem Definition
- → Algorithm Design / specification
- → Algorithm Analysis
- → Implementation
- → Testing and Debugging
- → Maintenance

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Algorithm Design / Specification / Strategy
  - → Divide and Conquer
  - → Greedy Approach
  - → Dynamic Programming
  - → Backtracking
  - → Branch and Bound.

- Algorithm Analysis

⇒ Many options - one select scenario (best option)

- → Space complexity : How much space needed?
- → Time complexity : How much time does it take?

>>> → Time > Space

→ Experimental Analysis

- Making of workable model (code) and Testing / Analysing it.
- Gives Exact analysis values
- But for same code we might get different values as it depends on H/W, S/W, PL, etc.

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

→ Absolute Analysis.

- Asymptotic Notations
- Only on Algorithm
- Order of Growth.

'O' 'Θ' Ω

$f(n) = O(g(n))$ [Upper bound]

$f(n) = \Theta(g(n))$ [Exact/Same]

$f(n) = \Omega(g(n))$ [Lower bound]

Example : "Bhavishyvani"

① → Life = 50 years

② → Life = 70 years

③ → Life = 30 years

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Examples:

TC	Type	Example
$O(1)$	Constant	Add an item to LL (front)
$O(\log n)$	Logarithmic	find an item in sorted array.
$O(n)$	Linear	find an item in unsorted array
$O(n^2)$	Quadratic	Nested loop (2)
$O(n^3)$	Cubic	Matrix multiplication
$O(2^n)$	Exponential	Fibonacci series

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Pseudcode

  - → High level Representation of an algorithm. (Human Understandable)
  - → Less formal (use of Natural language words / elements)
  - → NO specific syntax (like PL)
  - → Eg:-

```
Begin
Set Result = 0;
Input: num1, num2
Result = num1 + num2
End
```

- flowchart

  - → Diagram to show the flow of data
  - → Different shapes (▭, ⬭, ◇)
  - → Easy to understand
  - → Graphical Representation of Algo.
  - → Eg:

↓ continue.

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Eg:

- Small notations ('=' not allowed)
  - ↳ strict nature (less than or Greater than)

$f(n) = o(g(n))$ $[f(n) < g(n)]$

$f(n) = \omega(g(n))$ $[f(n) > g(n)]$

- "Recurrence" → (Recursion) [f^n calls itself]
  - ↳ Small problem, solve it
  - ↳ Large problem, divide it, solve it, Combine it.

* Recursion is expressed in → "Recurrences".

→ [when f^n calls itself, its running time is described/ given by "Recurrence eq^n"

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Eg:-

```
fn( arr, i, j, x)
    m = (i+j)/2
    if (arr[m] == x)
        return;
    else
        if (arr [m] > x)
            fn (arr, i, m-1, x)
        else
            fn (arr, m+1, j, x)
```

$$\boxed{T(n) = T(n/2) + c}$$

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Various methods / techniques to solve Recurrence

$$T(n) \begin{cases} T(n/2)+c & ; n>1 \\ 1 & ; n=1 \end{cases}$$

$$T(n) = T(n/2) + c$$

substitute $n = n/2$

$$T(n/2) = T(n/4) + c$$

$$T(n/4) = T(n/8) + c$$

$\therefore$ NOW Back substitute the things

$$T(n) = T(n/4) + c + c$$

$$= T(n/2^2) + 2c$$

also, same for subsequent eq^{n}s

$$= T(n/8) + 2c + c$$

$$= T(n/2^3) + 3c$$

$$\downarrow$$

$$T(n/2^4) + 4c$$

$$\downarrow \text{ k times.}$$

$$T\left(\frac{n}{2^k}\right) + kc$$

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Now we want $T\left(\frac{n}{2^k}\right)$ as '1'

So, sub: $2^k = n$

$$\therefore \quad T\left(\frac{n}{n}\right) + kC$$

$$= T(1) + kC$$

for $n = 1$
$\therefore T(1) = 1$

$k \log 2 = \log n$
$\therefore k = \log n$

$$\therefore \Rightarrow \quad 1 + \log n \cdot C$$

Reduce to

$$\Rightarrow \quad O(\log n)$$

"Master Theorem" → faster
→ few not All (RR)

Eg: $T(n) = T(n+5) + 1$

$a = 1 \quad b = 1$

SM ✓ MM ✗

$$T(n) = aT(n/b) + f(n) \quad \boxed{a \geq 1} \quad \boxed{b > 1}$$

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

n = size of input

a = no. of subproblems in recursion

n/b = size of each subproblem

$f(n)$ = Cost (divide (P) & combine (S))

<u>Problems/ limitations</u> :

- $T(n) = \cos n$
- $f(n) = 2^n$
- $a = 5n$
- $a < 1$

<u>Solution</u> (MM): $T(n) = n^{\log_b a}[X]$

$$X = \frac{f(n)}{n^{\log_b a}}$$

① n^p; $p > 0 \longrightarrow O(n^p)$

② n^p; $p < 0 \longrightarrow O(1)$

③ $(\log n)^q$; $q \geq 0 \longrightarrow \frac{(\log n)^{q+1}}{q+1}$

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Eg: $T(n) = T(n/2) + C$

$a = 1 \quad b = 2 \quad f(n) =$ constant

$\Rightarrow T(n) = n^{\log_b a} \; [X]$

$n^0 = 1 \qquad \dfrac{C}{n^{\log_2 1}} = \dfrac{C}{n^0} = C$

So; $T(n) = \underline{1 \cdot C}$ → go for ③

$\therefore \quad = (\log_2 n)^0 \cdot C$ [need to include it]

$= \dfrac{(\log_2 n)^{0+1}}{0+1} \cdot C$

$\boxed{= O(\log_2 n)}$

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Eg:- $T(n) = 3T(n/2) + n^2$

$a = 1 \quad b = 2 \quad f(n) = n^2$

$\therefore T(n) = n^{\log_2 1} \; [\times]$

$n^0 = 1 \qquad \dfrac{n^2}{n^{\log_2 1}} = \dfrac{n^2}{1} = n^2$

$= 1 \cdot O(n^2) \longrightarrow (1)$

$\boxed{= O(n^2)}$

Eg:- $T(n) = 8T\left(\frac{n}{2}\right) + n$

$a = 8 \quad b = 2 \quad f(n) = n$

$T(n) = n^{\log_2 8} \; [\times]$

$n^{\log_2 (2)^3} \qquad \dfrac{n}{n^{\log_2 8}} = \dfrac{n}{n^{\log_2 (2)^3}} = \dfrac{n}{n^3} = n^{-2}$

n^3

$= n^3 \cdot O(1) \longrightarrow (2)$

$\boxed{= O(n^3)}$

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- **Solvable**: Can be solved
  or
  It can be proved that 'P' cannot be solved.

- **Unsolvable**: Can't solve
  Can't Prove that it cannot be solved

- **Decidable**: Problem that can be solved
  + in finite time
  or
  Reasonable amt of time

- **Undecidable**: No definite answer for a problem in finite time

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- P type :- Polynomial type/Time
  $\rightarrow n^k \; [n^1, n^2, n^3 \ldots]$

- <u>NP type</u> : Non-Polynomial type/Time
  $\rightarrow n^n \; [2^n, 3^n \ldots]$

"NPC are hardest to solve in NP"

"NPC are easiest to solve in NPH"

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Searching & Sorting.
  - → Linear
  - → Binary
  - → Bubble
  - → Selection
  - → Insertion
  - → Merge
  - → Quick
  - → Counting
  - → Radix
  - → Bucket
  - → Heap.

- Linear Search → Simple
  - → Like name → Linear fashion

→ 5 4 2 1 9 $\boxed{x = 1}$

1 → 5 4 2 1 9 (x)

2 → 5 4 2 1 9 (x)

3 → 5 4 2 1 9 (x)

4 → 5 4 2 1 9 (✓)

Best case → Index '0'
→ $O(1)$

worst case = last index
→ $O(n)$

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Binary search (Already studied it)

Time Complexity : Best case = $O(1)$
worst Case = $O(\log n)$

- Bubble Sort

→

5	4	2	1	3

1) 5 4 2 1 3

⇒ 5 > 4 [swap]

∴ 4 5 2 1 3

2) 4 5 2 1 3

⇒ 5 > 2 [swap]

∴ 4 2 5 1 3

3) 4 2 5 1 3

⇒ 5 > 1 [swap]

∴ 4 2 1 5 3

4) 4 2 1 5 3

⇒ 5 > 1 [swap]

∴ 4 2 1 3 5

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

1) 4 2 1 3 5

4 > 2 [swap]

∴ 2 4 1 3 5

2) 2 4 1 3 5

4 > 1 [swap]

3). 2 1 4 3 5

4 > 3 [swap]

1) 2 1 3 4 5

2 > 1 [swap]

∴ 1 2 3 4 5

2) 1 2 3 4 5

2 < 3 [No swap].

```
for i = 1 to n
{
  k = 1
  while ( k ≤ n-i )
    { If ( a[k] > a[k+1] )
        { swap ( a[k], a[k+1] ) }
      k = k+1
    }
}
```

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Selection Sort

→

0	1	2	3	4
10	6	5	7	1

	0	1	2	3	4	
1)	10	6	5	7	1	min = '0'
2)	10	6	5	7	1	min = '1'
3)	10	6	5	7	1	min = '2'
4)	10	6	5	7	1	min = '2'
5)	10	6	5	7	1	min = '4'

swap

1)	1	6	5	7	10	min = '1'
2)	1	6	5	7	10	min = '2'
3)	1	6	5	7	10	min = '2'
4)	1	6	5	7	10	min = '2'

swap

1)	1	5	6	7	10	min = '2'
2)	1	5	6	7	10	min = '2'
3)	1	5	6	7	10	min = '2'

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

"Already in place"

1) 1 5 6 7 ⟷ 10 min = '3'

2) 1 5 6 7 10 min '3'

"Already in place"

<u>final</u> ⇒ 1 5 6 7 10

```
ss(arr, )
{
    for ( i=0; i< n-1; ++)
    {
        min= i
        for ( j= i+1 ; j< n ; j++)
        {
            if ( arr[j] < arr[min]
            min = j
        }
        swap( arr[min], arr[i])
    }
}
```

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Insertion Sort

	0	1	2	3	4
⇒	9	5	1	4	3

	0	1	2	3	4	
1)	9 ↔	5	1	4	3	index 1 ↓ key = 5
2)	9	9	1	4	3	key = 5
3)	5	9	1	4	3	(step 1 complete).

Step 2.

1)	5	9 ↔	1	4	3	index 2 ↓ key = 1
2)	5	9	9	4	3	key = 1
3)	5	5	9	4	3	key = 1
4)	1	5	9	4	3	(Step 2 complete)

Step 3

1)	1	5	9	4	3	index 3 ↓ key = 4
2)	1	5	9	9	3	key = 4
3)	1	5	5	9	3	key = 4
4)	1	4	5	9	3	(Step 3 complete)

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

step 4

index 4

1) 1 4 5 9 3 key = 3

2) 1 4 5 9 9 key = 3

3) 1 4 5 5 9 key = 3

4) 1 4 4 5 9 key = 3

5) 1 3 4 5 9 key = 3 placed (step 4 complete)

```
IS ( arr , n)
{
    for ( step= 1 to n )
    {
        key = arr [step]
        j = step-1

        while ( key < arr[i] && j ≥ 0)
        {
            arr[j+1] = arr[j]
            j = j -1
        }
        arr [j+1] = key
    }
}
```

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Counting Sort

Eg:- 5 3 2 2 6 4 1

① Max = 6 (6+1)

②

0	1	2	3	4	5	6

Count array

③

0	1	2	1	1	1	1
0	1	2	3	4	5	6

update the count values

④

0	1	3	4	5	6	7
0	1	2	3	4	5	6

update the cumulative sum

⑤

0	1	3	4	5	6	7
0	1	2	3	4	5	6

(1 - 1) (4 - 1) (5 - 1) (6 - 1) (7 - 1)

final o/p

1	2	2	3	4	5	6
0	1	2	3	4	5	6

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Eg:- CS (arr , n)

```
{
    max = arr[0]
    for ( i = 1 to n )
    {
        if ( arr[i] > max)
        max = arr[i]
    }
```

(1)

```
// Count arr size = max + 1
```

(2)

```
for ( j = 1 to n )
{
    Count [ arr [ j ] ] ++
    Count [arr[j]] = Count[arr[j]] + 1
}
```

(3)

```
for ( i = 1 to max )
{
    Count [i] += Count [i-1]
}
```

(4)

```
for ( i = n-1 to 0 )
{
    Out [ Count [arr[i] ] - 1] = arr[i]
    Count [ arr[i] ] --
}
```

(5)

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Radix Sort

Eg:- 222 153 654 20 751

'U' sorting

2	2	2
1	5	3
6	5	4
0	2	0
7	5	1
H	T	U

0	2	0
7	5	1
2	2	2
1	5	3
6	5	4
x	x	✓

'T' sorting.

0	2	0
2	2	2
7	5	1
1	5	3
6	5	4
x	✓	✓

'H' sorting.

0	2	0
1	5	3
2	2	2
6	5	4
7	5	1
✓	✓	✓

done.

Here, we sort the 'U', 'T', 'H' elements by using counting sort.

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Bucket sort

Eg:-

0.45 0.75 0.62 0.15 0.33

0.27 0.55 0.88 0.93 0.05

⇒ size = 10

{ $0.45 \times 10 = \lfloor 4.5 \rfloor = 4$ }

⇒ BS{arr, n}

{ New array A for size n

for i = 0 to n-1

{ A[i] empty list }

for i = 1 to n

{ insert arr[i] into list A[n × A[i]] }

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

for i = 0 to n-1

{ list A[i] will be sorted }

Then finally just concatenate the list.

- <u>Heap Sort</u>

<u>Complete BT</u>

Max Heap

Min Heap

Eg:- 30 40 20 35 50 45

CBT

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Max heap

Exchange Root with last element

0	1	2	3	4	5
20					50

0	1	2	3	4	5
30				45	50

0	1	2	3	4	5
			40	45	50

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

20		35	40	45	50
0	1	2	3	4	5

20	30	35	40	45	50
0	1	2	3	4	5

- Merge Sort

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

```
Mergesort ( arr, p, r)
  {
    if ( p < r )
      {
        m = ⌊(p+r)/2⌋
        mergesort ( arr, p, m)
        mergesort (arr, m+1, r)
        merge ( arr, p, m, r)
      }
  }
```

Downloaded by Sushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Quick sort

1	2	3	4	5	6
60	50	30	40	25	55

```
Partition (arr, p, r)
{
    V = arr[r]
    i = p - 1
    for j = p to (r-1)
    {
        if(arr[j] ≤ x)
        { i = i+1
          swap ( arr[i], arr[j])
        }
    }
    swap ( arr[i+1], arr[r])
    return i+1
}

QS ( arr , p, r)
{   if (p < r)
    {
        x = partition(arr, p, r)
        QS ( arr, p, x-1)
        QS (arr, x+1, r)
    }
```

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Algorithm	TC	SC
Linear Search →	$O(n)$	$O(1)$
Binary search →	$O(\log n)$	$O(1)$
Bubble sort →	$O(n^2)$	$O(1)$
Selection sort →	$O(n^2)$	$O(1)$
Insertion Sort →	$O(n^2)$	$O(1)$
Heap Sort →	$O(n \log n)$	$O(n)$
Bucket sort →	$O(n^2)$	$O(n)$
Radix sort →	$O(nk)$	$O(n+k)$
Merge sort →	$O(n \log n)$	$O(n)$
Quick sort →	$O(n^2)$	$O(\log n)$
Counting Sort →	$O(n+k)$	$O(k)$

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Divide & Conquer

→ Merge Sort
→ Quick Sort
→ Binary Search
→ Strassen's Algorithm

eg:-

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} (ae+bg) & (af+bh) \\ (ce+dg) & (cf+dh) \end{bmatrix}$$

$P1 = a(f-h)$
$P2 = (a+b)h$
$P3 = (c+d)e$
$P4 = d(g-e)$
$P5 = (a+d)(e+h)$
$P6 = (b-d)(g+h)$
$P7 = (a-c)(e+f)$

$$\longrightarrow = \begin{bmatrix} (P5+P4-P2+P6) & (P1+P2) \\ (P3+P4) & (P1+P5-P3-P7) \end{bmatrix}$$

$$\boxed{\begin{aligned} T(n) &= 8T(n/2) + n^2 \\ T(n) &= 7T(n/2) + n^2 \end{aligned}}$$

$\rightarrow O(n^3)$

$\rightarrow O(n^{\log_2 7}) \approx O(n^{2.807})$

This document is available free of charge on studocu
Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Convex Hull

(✓)

(×)

eg:

① select/start from the min y value (leftmost)

② Labeling the points anticlockwise

③

④

⑤

⑥

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

(7)

(8)

- Greedy Approach

⇒ Solve the problem with best optimal Solution. out of all feasible solutions

Note : 1) Optimal solution always (x)
2) Time required is reduced (Fast)
3) Look for local optimal soln at every step

Eg:- Activities : → (Activity seckeion Problem)

	A1	A2	A3	A4	A5	A6
ST :	1	1	3	4	4	5
ET :	2	3	4	5	6	7

Ans

(4) activities

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

```
Activities( S, E, n )
{
    i = 0
    for ( j = 1 to n )
    {
        if (S[j] ≥ E[i])
        {   print ( j )
            i = j
        }
    }
}
```

<u>Applications</u>: "Meet Scheduling"

- <u>Knap Sack Problem</u>
  - → fractional : fraction allowed
  - → 0/1 : All or None

<u>Eg</u>: Greedy Approach

(Profit)	(weight)	(Profit/weight)
→ more	→ less	→ more
5 ✓ 3 ✗	5 ✗ 3 ✓	5 ✓ 3 ✗

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Eg

Objects	1	2	3
P :	20	15	10
W :	15	10	5
$\frac{P}{W}$:	1.33	1.5	2

Max = 18

By Profit :

$$\overset{1}{20} + \overset{2}{15} \times \frac{3}{10} = 24.5$$

By weight :

$$\overset{3}{10} + \overset{2}{15} + \overset{1}{20} \times \frac{3}{15} = 29$$

By P/W :

$$\overset{3}{10} + \overset{2}{15} + \overset{1}{20} \times \frac{3}{15} = 29$$

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Job sequencing Problem (J & D)

↳ focus on 'Profit'

+

Execute it Late

Eg:- Jobs

Jobs	J_1	J_2	J_3	J_4	J_5	J_6
P :	40	35	30	25	20	15
D :	3	3	4	2	1	3

1) $J_1 \rightarrow \{40, 3\}$

		J1	
1	2	3	4

2) $J_2 \rightarrow \{35, 3\}$

	J2	J1	
1	2	3	4

Profit
$= 40 + 35 + 30 + 25$
$= 130$

3) $J_3 \rightarrow \{30, 4\}$

	J2	J1	J3
1	2	3	4

4) $J_4 \rightarrow \{25, 2\}$

J4	J2	J1	J3
1	2	3	4

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Huffman Coding

→ Prefic Code/sequence : A 'S' cannot be a Prefix of any other 'S' in give sequence set.

eg:- {00, 11, 01, 10}

{00, 001, 110, 1100}

Eg

A	B	C	D	E	F
10	8	12	16	5	2

1) Ascending order

2 5 8 10 12 16

2) Combine first 2

← place it in the above order

8 10 12 16

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

3)

→ place it.

→ 4)

5)

Ⓐ : 0 0

Ⓑ : 1 0 1

Ⓒ : 0 1

Ⓓ : 1 1

Ⓔ : 1 0 0 1

Ⓕ : 1 0 0 0

weight

$= 10 \times 2 + 8 \times 3 + 12 \times 2 + 16 \times 2 + 5 \times 4 + 2 \times 4$

$= 128$

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Dijkstra's Algorithm

→ Single Source shortest Path.

eg:-

Relaxation: If $d(u) + c(u,v) < d(v)$

then $d(v) = d(u) + c(u,v)$

start from 'A'

$$d(A) + c(A, B)$$
$$0 + 10 = 10 < \infty$$

$\therefore d(B) = 10$

$$d(A) + c(A, C)$$
$$0 + 30 = 30 < \infty$$

$\therefore d(C) = 30$

$$d(B) + c(B, C)$$
$$10 + 5 = 15 < 30$$

$\therefore d(C) = 15$

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Eg

Once 'v' relaxes check if any edge exists that give better 'd'

$S_{①}$	2	3	4	5
1	2	∞	3	∞
1,2	2	9	3	11
1,2,4	2	9	3	11
1,2,4,3	2	9	3	11
"1,2,4,3,5"				

$1 \to 2$ $1 \to 3$ $1 \to 4$ $1 \to 5$

-reweight Problem

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Bellman ford Algorithm (v-1) times Relax each E

DA: 1 → 3 : 2

Now Relax 1 more time to check -ve cycle.

(hence NO -ve weight cycle)

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

-ve WC Exists

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Dynamic Programming

- → Like D & C
- → Subproblems
- → Solutions (solve once & save)
- → NO recomputing
- → memoization

Eg fibonacci series

$f(0) = 0$

$f(1) = 1$

$f(n) = f(n-1) + f(n-2)$

⇒ 0 1 1 2 3 5 8

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Top-down Approach

- 0/1 knapsack (using Greedy)
  - 0 (Nothing)
  - 1 (Everything)

Obj	1	2	3
P	20	25	60
W	2	4	8
$\frac{P}{W}$	10	6.2	7.8

Max = 12

But $2 + 8 = 10$

Available $= 12 - 10$
$= 2$

$\Rightarrow$ 1 3

$20 + 60 = 80$

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

◦ Recurrence Equation

$0/1\ KP[n-1], [W-W_a] + P_a$ } Included / (P)

$0/1\ KP[n-1], W$ } Excluded / (A)

$\rightarrow$ $[n-1], m$ $\{W_a > W\}$

$\rightarrow$ max ((P), (A))

$\rightarrow$ Considering all $(O(2^n))$

$\rightarrow$ Considering DP $(O(n \cdot w))$

Initially = W = 0

P	W	W = 1	W = 2	W = 3
1	1	P = 1	P = 1	P = 1
1	1	P = 1	P = 2	P = 2
1	1	P = 1	P = 2	P = 3

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Floyd Warshall Problem

Every vertex to Every other vertex

Shortest Distance

[All pair shortest Path]

(A) ← 'x' Intermidiate nodes → (B)

So here
we take each 'v'
as an IN one by one

Eg:-

① If have self loops or parallel edges remove them.

① Direct edge.

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 8 & \infty & 1 \\ B & \infty & 0 & 1 & \infty \\ C & 4 & \infty & 0 & \infty \\ D & \infty & 2 & 9 & 0 \end{array}$$

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

2) 'A' as IN

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 8 & \infty & 1 \\ B & \infty & 0 & 1 & \infty \\ C & 4 & 12 & 0 & 5 \\ D & \infty & 2 & 9 & 0 \end{array}$$

3) 'B' as IN

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 8 & 9 & 1 \\ B & \infty & 0 & 1 & \infty \\ C & 4 & 12 & 0 & 5 \\ D & \infty & 2 & \cancel{9}\,3 & 0 \end{array}$$

4) 'C' as IN

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 8 & 9 & 1 \\ B & 5 & 0 & 1 & \cancel{\infty}\,6 \\ C & 4 & 12 & 0 & 5 \\ D & \cancel{13}\,7 & 2 & 3 & 0 \end{array}$$

B → C → A → D

D → B → C → A

5) 'D' as IN

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 3 & \cancel{8}\,4 & 1 \\ B & 5 & 0 & 1 & 6 \\ C & 4 & \cancel{12}\,7 & 0 & 5 \\ D & 7 & 2 & 3 & 0 \end{array}$$

A → D → B → C

C → A → D → B

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

eg:- $M0 =$

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 8 & \infty & 1 \\ B & \infty & 0 & 1 & \infty \\ C & 4 & \infty & 0 & \infty \\ D & \infty & 2 & 9 & 0 \end{array}$$

focus on '∞'

$M1 =$

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 8 & \infty & 1 \\ B & \infty & 0 & 1 & \infty \\ C & 4 & 12 & 0 & 5 \\ D & \infty & 2 & 9 & 0 \end{array}$$

$M2 =$

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 8 & 9 & 1 \\ B & \infty & 0 & 1 & \infty \\ C & 4 & 12 & 0 & 5 \\ D & \infty & 2 & 3 & 0 \end{array}$$

$M3 =$

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & \cancel{2}\,8 & 9 & \cancel{4}\,1 \\ B & 5 & 0 & 1 & 6 \\ C & 4 & 12 & 0 & 5 \\ D & 7 & \cancel{5}\,2 & 3 & 0 \end{array}$$

$M4 =$

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 3 & 4 & 1 \\ B & 5 & 0 & 1 & 6 \\ C & 4 & 7 & 0 & 5 \\ D & 7 & 2 & 3 & 0 \end{array}$$

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Sum of Subsets Problem

→ Given set = { 1, 3, 5, 9 }

Sum = 8

O/P → Y/N T/F

	1	2	3	4	5	6	7	8
1	T	F	F	F	F	F	F	F
3	T	F	T	T	F	F	F	F
5	T	F	T	T	T	T	F	T
9	T	F	T	T	T	T	F	T

{ 1 } { 1, 3 } { 1, 3, 5, 9 }

'1'

⇒ Set → 'n' elements

Sum = 'S'

→ [n-1, S - Wn]

→ [n-1, S]

→ [n-1, S] if Wn > S

$O(2^n)$

↓

$\Theta(n.S)$

(3, 3)

(2, 2) (2, 3)

(1, 1) (1, 2) (1, 2) (1, 3)

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

→ Travelling Salesman Problem

- Start from 's' visit every other vertex of graph, then come back to 's' [Hamiltonion Graph]
  ↳ with min dist/cost.

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Eg:

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 10 & 15 & 20 \\ 2 & 5 & 0 & 25 & 10 \\ 3 & 15 & 30 & 0 & 5 \\ 4 & 15 & 10 & 20 & 0 \end{array}$$

→ Brute force method

worst

$O(n!)$

$\boxed{n^n}$

Greed → 55

TSP(1, [2, 3, 4])

(55)

(10) (1,2) + TSP[2, [3,4]] (45)

(35)

(15) (1,3) + TSP[3, [2,4]] (20)

(60)

(20) (1,4) + TSP[4, [2,3]] (50)

(45) (45)

(5) (3,3) + TSP(3,4) (20)

(10) (2,4) + TSP (4,3) (35)

(55) (20)

(30) (3,2) + TSP(2,4) (25)

(5) (3,4) + TSP(4,2) (15)

50 (55)

(10) (4,2) + T(3,3) (40)

(20) (4,3) + T(3,2) (35)

(3,4) + (4,1)
(5) (15)

(4,3) + (3,1)
(20) (15)

(2,4) + (4,1) (10)
(15)

(4,2) + (2,1) (10) (5)

(2,3) + (3,1) (25) (15)

(3,2) + (2,1) (30) (5)

$1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 1 = \underline{35}$

Total $n \cdot 2^n$ sub problems → so $O(n^2 \cdot 2^n) < O(n!)$

This document is available free of charge on studocu
Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Backtracking

  ↳ Brute force method + Condition

  if fail → Backtrack (don't go ahead)
  if success → Go ahead.

Eg.

[vehicle can't go through H, S]

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Graph Colouring

⇒ Color all 'V' of 'G' such that 'NO' 2 adj. vertices have same color.

+

Minimum no. of colours.

Eg

(RG)

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

N-Queen Problem

(4) Queens

No conflict/attack

→ Row
→ column
→ diagonally.

State space tree

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Hamiltonian Cycle

$\Rightarrow \quad 1 \to 2 \to 3 \to 4 \to 1$

$1 \to 2 \to 3 \to 4 \to \overset{X}{2}$

Eg

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Branch & Bound Algorithm

↓ Generating subproblems

↓ Not Extending P. solutions that are expensive than current soln.

→ Extend the cheapest soln/path

→ "Minimization"

Eg

Min = 9

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

0/1 Knapsack Problem (B & B)

Obj	O_1	O_2	O_3	O_4	Max
P	10	20	30	40	10
W	4	6	3	1	

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Spanning Tree

1. It covers all 'v's
2. No cycle
3. Has total $(n-1)$ edges → no. of 'v's.

Graph ↓

STs ↓

✓

✗

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

- Prim's algorithm & Kruskal algorithm

eg:

eg:

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

→ Hashing

(Time ↓ Space ↑)

→ Key
→ Location
→ Hash f^n
  → Easy
  → Fast
  → low

→ Hash table.

eg: $f^n(H(k)) = k \bmod n$

$f^n(H(k)) = k \bmod n+1$

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

keys → (10, 11, 12, 13, 14, 15, 16, 17, 18, 19)

f^n : K mod n

n = 10

	I
10	0
11	1
12	2
13	3
14	4
15	5
16	6
17	7
18	8
19	9

HT

- Mid Square

⇒ 12 → square → 144

I = 4 (store 12)

⇒ 11 → square → 121

I = 2 (store 11)

- K = 123456 (folding method)

123 + 456

$$\begin{array}{r} 123 \\ +456 \\ \hline 579 \end{array}$$ mod n

∴ n = 10, so 579 mod 10

⇒ 9 (store 123456)

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Collision

CRT

- (Open H) chaining
- (Closed H)
  - Linear Probing
  - Quadratic Probing
  - Double Probing.

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Chaining

$k \rightarrow (12, 42, 55, 35, 16, 26)$

Linear Probing

k mod 10

$k \rightarrow (12, 42, 33, 44, 45)$

$h(k) \rightarrow$ 2, 2, 3, 4, 5

$k = 42$

$h(k) = 2$ (collision)

$i = 1$

$h(k) + i = 2 + 1 = \underline{3}$

Problems

↳ Primary clustering

Search Time ↑

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Quadratic Probing

$h(k) = k \bmod 10$

46	0
31	1
22	2
43	3
	4
	5
26	6
37	7
28	8
	9

$k \rightarrow 22, 26, 31, 43, 28, 37, 46, 52$

$\Rightarrow (h(k) + i^2) \bmod 10$

$46 \bmod 10 = 6$ [Collision]

$i = 1$

$(6 + (1)^2) \bmod 10 = 7$

$i = 2$

$(6 + (2)^2) \bmod 10 = 0$

k_1	k_2	k_3
2	2	2
3	3	3
6	6	6
1	1	1
8	8	8
7	7	7

(Secondary Clustering)

for <u>52</u>

$52 \bmod 10 = 2$ [Collision]

$i = 1$

$(2 + (1)^2) \bmod 10 = 3$

$i = 2$

$(2 + (2)^2) \bmod 10 = 6$

$i = 3$

$(2 + (3)^2) \bmod 10 = 1$

$i = 4$

$(2 + (4)^2) \bmod 10 = 8$

$i = 5$ $(2 + (5)^2) \bmod 10 = 7$

$i = 6$ $(2 + (6)^2) \bmod 10 = 8$

$i = 7$ $(2 + (7)^2) \bmod 10 = 1$

$i = 8$ $(2 + (8)^2) \bmod 10 = 6$

$i = 9$ $(2 + (9)^2) \bmod 10 = 3$

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)

Double Hashing

$$(h1(k) + i\,h2(k)) \% n$$

$h1(k) = k \bmod 13$

$h2(k) = 1 + (k \bmod 11)$

	0
79	1
	2
	3
69	4
	5
	6
98	7
72	8
	9
	10
	11
	12

$k \rightarrow 79, 69, 98, 72$

$h1(79) = 79 \% 13 = 1$

$h1(69) = 69 \% 13 = 4$

$h1(98) = 98 \% 13 = 7$

$h1(72) = 72 \% 13 = 7$ [Collision]

$\Rightarrow i = 1$

$[7 + 1 * (1 + 72 \% 11)] \% 13$

$\hookrightarrow 1$ [collision

$\Rightarrow i = 2$

$[7 + 2 * (1 + 72 \% 11)] \% 13$

$\hookrightarrow 8$ ✓

This document is available free of charge on studocu

Downloaded by Shushant Kumar Awasthi (sushantawasthi000@gmail.com)