**Amey Thakur B-50**

Terna Engineering College

Computer Engineering Department

Class: SE – B(2019 – 2020) Subject: Analysis of Algorithms

Assignment No. 2

**Q.1:A:** Explain difference between Dynamic Programming Approach and Greedy Methods.

**B:** Find the feasible solution for the following jobs using Job Sequencing with Deadline.

Let n = 4, (p1, p2, p3, p4) = (100, 10, 15, 27) and (d1, d2, d3, d4) = (2, 1, 2, 1).

**Q. 2:A:** What is least-cost search in 15 - puzzle problem?

B: Solve the sum of subset for the following data using backtracking.

N = 4, (w1, w2, w3, w4) = (11, 13, 24, 7) M = 31.

**Q. 3:** Explain different String matching algorithms.

Amey Thakur B-50

PAGE No.
DATE / /

Assignment 5

Q1. Differentiate between greedy methods and dynamic approach.

| Dynamic Programming | Greedy Approach |
|---|---|
| ① It guarantees optimal solution. | ① It does not guarentee an optimal solution |
| ② Subproblems overlap | ② Subproblems don't overlap |
| ③ It does more work | ③ It does little work |
| ④ It considers the future choices | ④ Only considers the current choices |
| ⑤ There is no specialized set of feasible solutions | ⑤ Construct the solution from set of feasible solutions |
| ⑥ Select choice which is globally optimum | ⑥ Select choice which is locally optimum |
| ⑦ Employ memorization | ⑦ There is no concept of memorization |

Amey Thakur B-50

Q3. Explain job sequencing with deadline

Ans:

- There are 'n' jobs to be processed on a machine
- Each job 'i' has a deadline $d_i \geq 0$ and profit $p_i \geq 0$.
- Profit is earned if and only if the job is completed by its deadline
- The job is completed if it is processed on a machine for a unit time
- Only one machine is available for processing jobs
- Only one job is processed at a time on machine
- A feasible solution is a subset of jobs J such that each job is completed by its deadline
- An optimal solution is a feasible solution with maximum profit value.

Amey Thakur B-50

PAGE No. / DATE / /

$n = 4$

| Profit | $P_1$ | $P_2$ | $P_3$ | $P_4$ |
|---|---|---|---|---|
| | 100 | 10 | 15 | 27 |
| Deadline | $D_1$ | $D_2$ | $D_3$ | $D_4$ |
| | 2 | 1 | 2 | 1 |

| | No. | Processing of | Allocated units | Remark | Profit |
|---|---|---|---|---|---|
| 1) | $J_1$ | 1 | [0-1] | Completed | 100 |
| 2) | $J_2$ | 2 | [0-1] | Completed | 10 |
| 3) | $J_3$ | 3 | [0-1] | Completed | 15 |
| 4) | $J_4$ | 4 | [0-1] | Completed | 27 |
| 5) | $J_1, J_2$ | (2, 1) | [0-1] [1-2] | Completed | 110 |
| 6) | $J_1, J_3$ | (1, 3) | [0-1] [1-2] | Completed | 115 |
| 7) | $J_4, J_1$ | (4, 1) | [0-1] [1-2] | Completed | 127 |
| 8) | $J_2, J_3$ | (3, 2) | [0-1] [1-2] | Completed | 25 |
| 9) | $J_2, J_4$ | – | – | Rejected | – |
| 10) | $J_3, J_4$ | (4, 3) | [0-1] [1-2] | Completed | 42 |

In all above calculated feasible soln. soln no. 7 i.e. [4, 1] give max profit of 127

∴ Soln - 7 is optimal soln.

Amey Thakur B-50

Ans (2)

A] i) Each node in state space tree is associated with some cost. Cost function is applied to each node and function helps to select the next E node. E node is the node being evaluated. The node with min cost should be selected for further expansion

ii) The cost function is defined as $f(x) = g(x) + h(x)$, where $g(x)$ is the cost of reaching to current state from the initial state and $h(x)$ is the cost of reaching from current state to answer state

iii) Cost function for 15 puzzle is defined as the no of tiles on wrong location

iv) $c(x) = f(x) + h(x)$, where $f(x)$ is the length from the root node in state space tree and $h(x)$ is the non-blank tiles which are not on the correct location

v) For the initial state, there are four possible moves (left, right, up, down) the cost for all 4 is shown

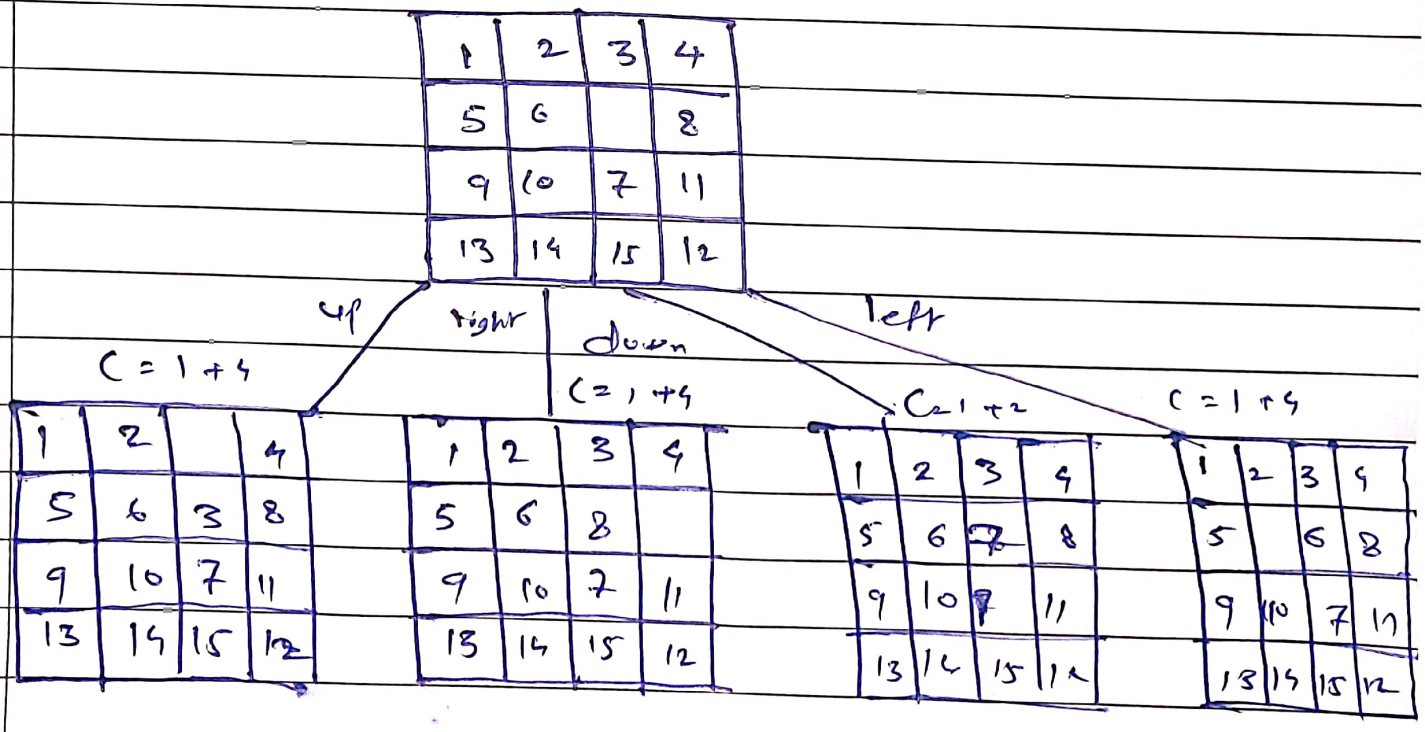



Fig Cost for 4 children after first move

Amey Thakur B-50 __/__/____

vi) 3rd configuration here has the minimum (least) cost, so remaining three states will be cut down and not explored further.

vii) This is the least cost state of a 15 puzzle problem

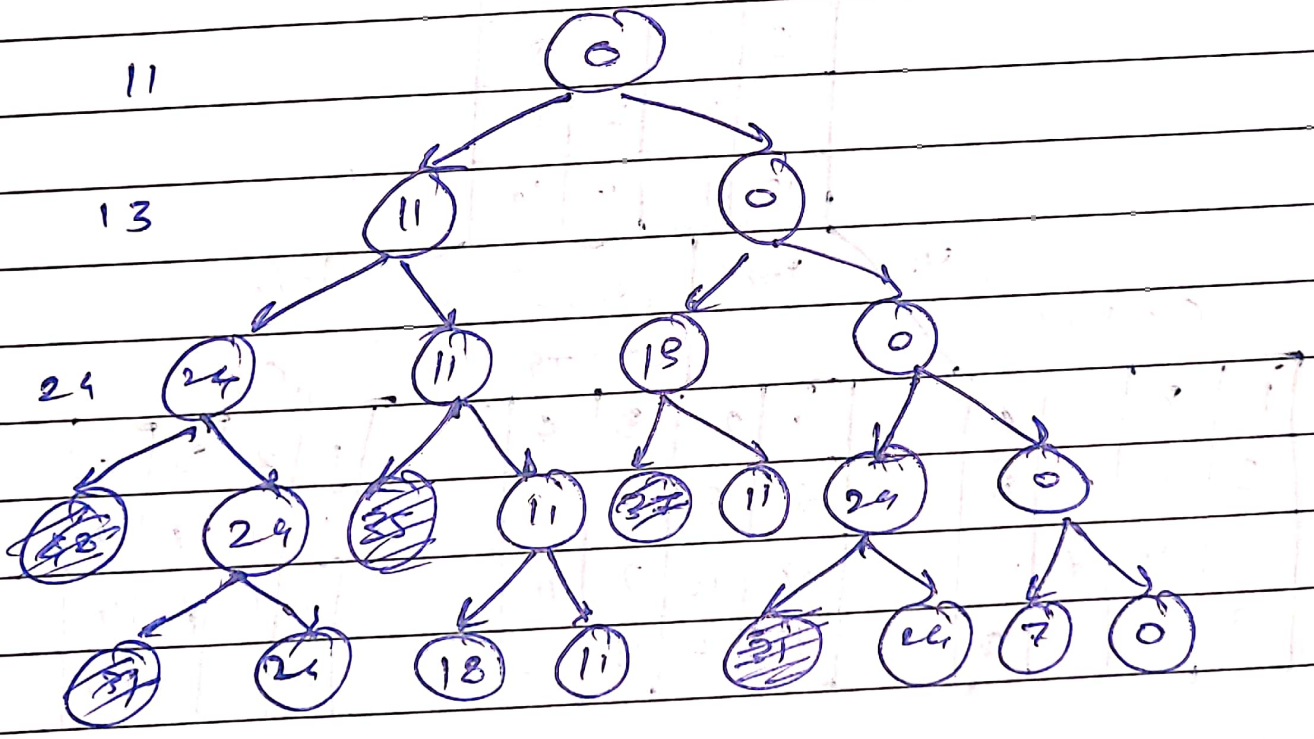
Amey Thakur B-50

Ans (2) [B]

$(w_1, w_2, w_3, w_4) = (11, 13, 24, 7)$ & $M = 31$, $n = 4$

| Items in subset | Condition | Comment |
|---|---|---|
| { } | 0 | Initial condition |
| {11} | $11 < 31$ | Add next element |
| {11, 13} | $24 < 31$ | Add next element |
| {11, 13, 24} | $48 < 31$ | Add set sum exceeds, so backtrack |
| {11, 13, 7} | 31 | Soln found |

- State space tree



Amey Thakur B-50

__/__/____

In the graph, the crossed circles show the result. Gray nodes show from where the algorithm backtracks. No. in the leftmost column indicates element under consideration at that level. Left & right branch represent inclusion and exclusion of that element respectively.

We get 2 sol^n.

(1) {11, 13, 7}

(2) {24, 7}

Amey Thakur B-50

Ans (3)

There are many types of String matching algorithm some of them are explained as follows.

(i) The naive string matching algorithm

- This is a simple & inefficient brut force approach It compares 1st character of pattern with searchable text. If match is found, pointers in both strings are advanced. If match is not found, pointer of text is incremented and pointer of pattern is reset. This process is repeated till the end of text.
- Naive approach does not require any pre processing

```
NAIVE_STRING_MATCHING (T, P)
for i ← 0 to n-m do
    if P[1,....m] = = T[i+1,....i+m] then
        print "Match Found."
    end
end
```

Amey Thakur B-50

(ii) The Robin Karp Algorithm

- Comparing no. is easier and cheaper than comparing strings. Robin Karp Algorithm represents string in number.
- Suppose p represents value corresponding to pattern p[1...m] of length m. And ts represent value of m-length substrings T[(s+1)...(s+m)] for s = 0, 1, 2, ..., n-m.
- We can compute p in O(m) time and all ts can be computed in O(n-m+1) time.
- Robin Karp Algorithm is based on hashing technique

```
RABIN_KARP (T, P)
h ← powers (d, m-1) mod q
p ← 0
to ← 0
for i ← 1 to m do
    p ← (d * p + P[i]) mod q
    to ← ((d * to) + T[i]) mod q
end

for s ← 0 to n-m do
    if p = = ts then
        if P[1...m] = = T[s+1 ... s+m] then
            print "Match found at shift", s
        end
    end
    if s < (n-m) then
        ts+1 ← (d * (ts - T[s+1]) * h) + T[s+m+1]
    end
end
```

Amey Thakur B-50

iii) String matching with finite automata.

- The idea of this approach is to build finite automata to scan text T for finding all occurences of pattern P.
- This approach examines each character of text exactly once to find the pattern.

Thus it takes linear time for matching but preprocessing time may be large.

Finite automata is defined by Tuple

$$M = \{Q, \Sigma, q_0, F, \delta\}$$

```
FINITE_AUTOMATA (T, P)
  State ← 0
    for i ← 1 to n do
    state ← δ (state, ti)

if  state = = m  then
    print "match found at position", i - m + 1
end
end
```