HW Assignment-2

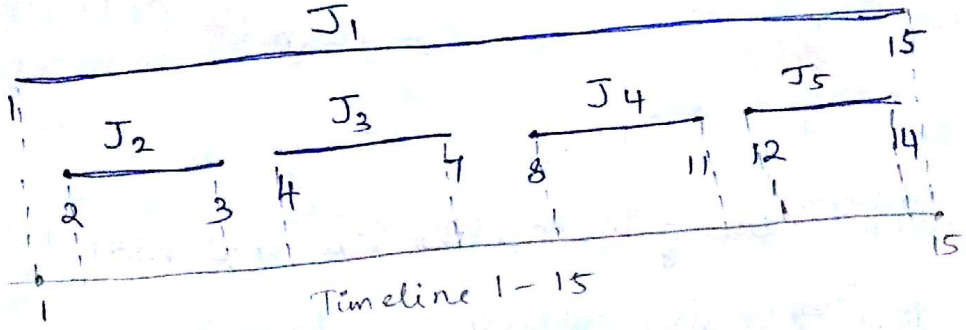
Dhayalini Nagaraj
A20359686

1. There are alternatives to CLRS3's greedy choice of the earliest finishing time. For each of the following alternatives, prove or disprove that it constructs an optimal schedule and briefly outline the implementation costs.

a) Job that ends latest

Let us consides five jobs and lets denote it by J_1, J_2, J_3, J_4 and J_5

Let the timeline be 1 – 15



DisProof:

According to Greedy, choose the job that ends latest will choose the first job J_1 (1, 9) then we would have scheduled only 1 job in the time frame of 1 – 15 but the

optimal solution is to do all the other four jobs J_2, J_3, J_4, J_5:

Disproof is by providing an example of five jobs.

According to start time, Finish time the jobs can be arranged in the order like (1,15), (2,3), (4,7), (8,11), (12,14).

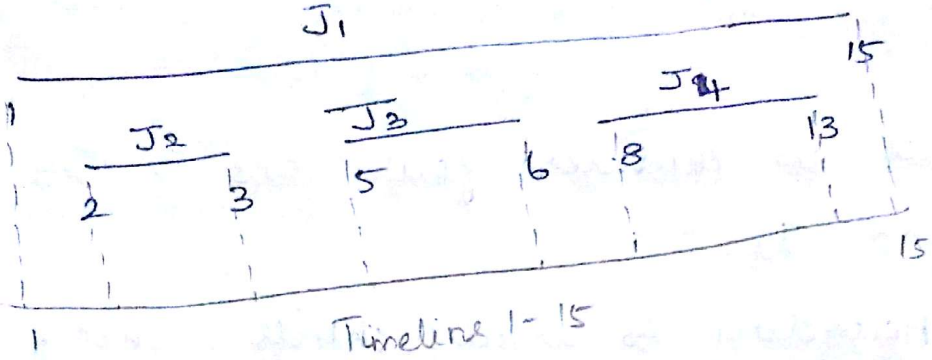
The greedy algorithm will give the solution as (1,15). If the algorithm finds an optimal solution then there should not have any non conflicting intervals higher than this.

But, there is another solution (2,3), (4,7), (8,11), (12,14).

Hence "Choose the job that ends latest" does not give optimal solution.

1.b). Jobs that starts first.

Let us consider four jobs and let's denote it by J_1, J_2, J_3, J_4



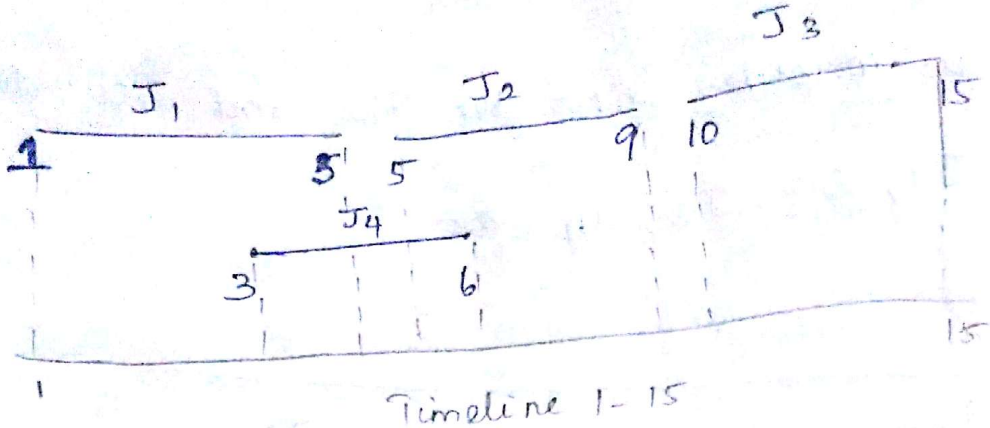
Let us disprove this. From given solution we can prove that "Choose the jobs that starts first" will not provide an optimal solution. Let us arrange the jobs in an order according to start time and finish time The jobs are ordered accordingly (1, 15), (2, 3) (5, 6), (8, 13).

According to the greedy choice jobs that starts first will give (1, 15) as the solution.

But the optimal solution is (2, 3), (5, 6), (8, 13).

Hence according to greedy choice "choose the jobs that starts first" will not provide an optimal solution.

1.c) Job of shortest duration. 2)



Timeline 1-15

Let us consider four jobs J_1, J_2, J_3 and J_4.

According to greedy choice, Choose the job of shortest duration will give the solution as (3,6)(10,5).

Let us consider the job intervals (1,5), (3,6), (5,9), (10,15).

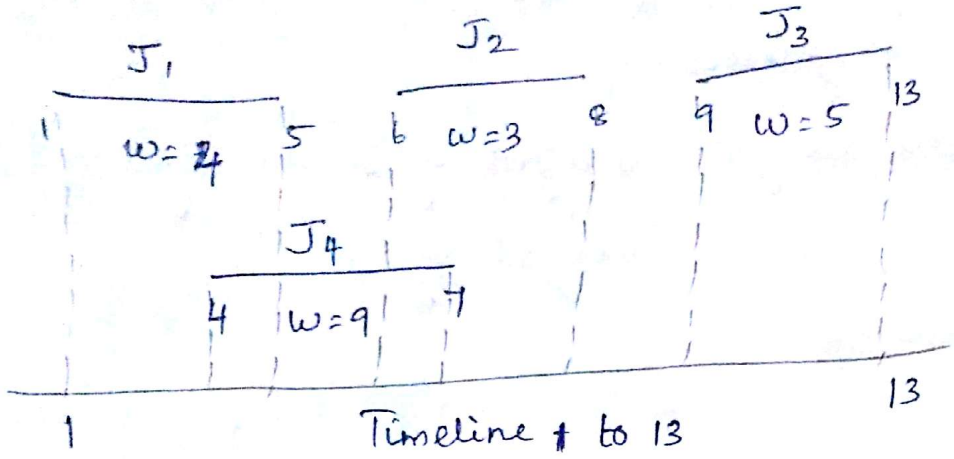
The optimal solution is (1,5), (5,9), (10,15)

In this, within 1 to 15 time frame three scheduled jobs can be completed.

But in greedy choice only two jobs can be completed J_4 and J_3.

Hence according to greedy choice "Choose the job of shortest duration" will not provide optimal solution.

2a) Prove that the heuristic of choosing the earliest finishing time does not always result in an optimal schedule



Weighted form of activity selection involves weight for every job. The heuristic of choosing the earliest finishing time that gives us the optimal answer should give a non conflicting subset of jobs that has maximum possible weight.

Let us consider four jobs J_1, J_2, J_3 and J_4.

Weight of $J_1 = 4$

Weight of $J_2 = 3$

Weight of $J_3 = 5$

Weight of $J_4 = 9$

Earliest finishing time heuristic will give the solution as (1,5), (6,8), (9,13). That is J_1, J_2 and J_3 can be scheduled and completed.

Total weight = weight of J_1 + weight of J_2 + weight of J_3

$= 4 + 3 + 5$

$= 12.$

The optimal solution is (4,7), (9,13).

(ii) J_4 and J_3 can be scheduled and completed.

Total weight = weight of J_4 + Weight of J_3

$= 9 + 5$

$= 14$

Job J_4 has the maximum weight and and the next non conflicting Job is J_3. So the heuristic of choosing the earliest finishing time does not always result in an optimal solution.

2b. Find an $O(n^2)$ to compute an optimal schedule.

Suppose there are n given jobs.
Let job i has a start time st_i and finish time ft_i and weight w_i.

Let us find a set S of compatible jobs whose total weight is maximized.

Assume n jobs are ordered in non decreasing order.
Let $S(j)$ be the maximum weight of set of compatible jobs.

The job with largest index less than j and the job which is compatible with job j is $p(j)$.

~~(a) largest k~~,

Case 1:
Let set S select job j

* It cannot use incompatible jobs $\{p(j)+1, p(j+2), \ldots, j-1\}$
* It must include optimal solution to problem consisting of remaining compatible jobs $1, 2, \ldots p(j)$.

case 2: S does not select job j

It must include optimal solution to problem consisting of remaining compatible jobs $1, 2, \cdots, j-1$

$$\therefore S(j) = \begin{cases} 0 & \text{if } j = 0 \\ \max S(j-1), w_i + S(p(j)) \end{cases}$$

$S(j)$ - optimal

compute $p(i)$ for i from 1 to n

Then the algorithm goes as.

set $S(0) = 0$ and $S(1) = w_1$

loop over i from 2 to n

set $S(i) = \max\{S(i-1), w_i + S(p(i))\}$

end loop.

Sorting takes $O(n \log n)$ and computation of $p(i)$ takes $O(n \log n)$. This is the optimal schedule.

Using linear search, the total running time is $O(n^2)$.

Let us use another set A of jobs and opti

The jobs are sorted according to increasing finish time

compute $p(i)$ from i to n

set $S(0) = 0$ and $S(1) = w_1$

set $A(0) = 0$ and $A(1) = \{1\}$

loop over i from 2 to n

if $S(i-1) > w_i + S(p(i))$

set $A(i) = A(i-1)$ and $S(i) = S(i-1)$

else $S(i-1) \leq w_i + S(p(i))$

set $A(i) = \{i\} \cup A(p(i))$ and $S(i) = w_i + S(p(i))$

end if

end loop.

This algorithm total running time is $O(n^2)$ and uses $O(n^2)$ space.

3. a) Describe, prove correct and analyze a greedy algorithm to find the minimum number of refueling stops needed to complete your trip. Prove that the algorithm is correct.

According to the problem, we need to find minimum number of refueling stops to complete the trip.

The car fuel stored in a battery must be replaced after at most 100 miles.

(ie) If the distance between the fuel station is more than 100 miles or if remaining fuel cannot cover the distance between two fuel station, then we must refuel it.

Algorithm for Minimum no. of refuelling stops

Let the initial stop be set to 1 and last stop be n

D[i] is the distance

C[i] is the cost

D[1.... n] Let us consider initial, next and

```
minstop ← 1                    // Let initial distance
i ← 1                              and stop be i
while i < n do                 // Last stop is n
    next ← i                   // Let the next stop be
                                   next
    while next < n and 100 ≥ D[next+1] - D[i] do
        next ← next + 1
    end while
    i ← next
    if i = n then
        return minstop
    end if
    minstop ← minstop + 1
end while
```

Proof: Let us assume the solution given by the above algorithm is MIN[1....k]. Let the optimal solution be MINOPTIMAL[1....n]. Bothe the solutions will store the refueling station ID's. Let us take one stop or first stop as 'initial'. Then MIN[initial] is different from MINOPTIMAL[initial]

According to greedy algorithm, refueling will be done as late as possible.

so $D[\text{MINOPTIMAL}[\text{initial}]] < D[\text{MIN}[\text{initial}]]$.

We can use exchange of parameters and induction techniques to ~~process~~ prove the greedy algorithm

Exchanging Arguments work by showing the we can ~~transform~~ iteratively transform any optimal solution into solution produced by greedy. Define solution and compare solutions, exchange the pieces and then iterate it. During iteration process we can prove it using induction.

After making greedy first choice, we are left with smaller subproblem which will also be optimal. (ie) MINOPTIMAL' is also an optimal solution.

Let MINOPTIMAL' be a new optimal solution and it will have same refuelling stops like MINOPTIMAL. ~~because~~ MIN[initial]. The MINOPTIMAL' is formed by replacing MINOPTIMAL[initial] by MIN[initial].

MINOPTIMAL' will have same stops like MINOPTIMAL because MIN[initial] in MINOPTIMAL can reach MINOPTIMAL[initial+1].

Again replace MINOPTIMAL by MINOPTIMAL'.

So we should follow the similar process recursively by comparing MIN and MINOPTIMAL.

Replace MINOPTIMAL[initial] by MIN[initial] if MIN[initial] is different from MINOPTIMAL[initial]

(ie) MIN[initial] ≠ MINOPTIMAL[initial]

the replace or exchange the pieces.

By doing this process iteratively we can get MINOPTIMAL' which is same as MIN which has same number of equal stops as MINOPTIMAL.

(ii) It denotes MIN has same number of equal stops as MINOPTIMAL.

Hence it is proved.

3b) Show that your greedy algorithm in part (a) does not minimize the total cost of travel.

Assume $n = 4$.

Let $D[i]$ be the distance covered and

$C[i]$ be the cost.

$D[1] = 0$ $D[2] = 15$ $D[3] = 70$ $D[4] = 103$

$c[1] = 0$ $c[2] = 8$ $c[3] = 100$ $c[4] = 0$

Using greedy algorithm, refueling must be done in station 3 to reach the destination so the total cost is $c[1] + c[3]$.

$$c[1] + c[3] = 0 + 100 = 100.$$

But the optimal solution is $c[1] + c[2]$.

$c[1] + c[2] = 0 + 8 = 8$. The destination can be reached and total cost is minimum.

3c) Describe an efficient algorithm to compute the locations of the fuel stations you should stop at that does minimize the total cost of travel.

Dynamic Programming can be used.

mincost(i,j) be the minimum cost to arrive station 'i', but the car can still go to j miles.

We can either refuel it at i station or we can continue to next station without filling fuel.

$$mincost(i,j) = \begin{cases} mincost(i-1, j+D[i]-D[i-1]) & 1 \le i \le n \cap 0 \le j \le 100 \cap j + D[i] - D[i-1] \le 100 \\ \min(mincost(i-1,k)) + C[i] & 1 < i \le n \cap j = 100 \cap k \ge D[i] - D[i-1] \end{cases}$$

Algorithm:

Let us assume mincost elements to be ∞

D[i] distance, C[i] cost, n - last station,

i - arrive station, j - remaining miles.

```
mincost(1, 100) = C[1]
for i = 2 to n do
    for j = 0 to 99 do
        if j + D[i] - D[i-1] ≤ 100 then
            mincost(i,j) = mincost(i-1, j+D[i] - D[i-1])
        endif
    endfor
```

```
for k = 0 to 100 do
    if k ≥ D[i] - D[i-1] then
        mincost(i, 100) = min(mincost(i,100), mincost(i-1,k) + C[i])
    end if
end for
end for
cost = ∞
for j = 0 to 100 do
    cost = min(cost, mincost(n, j))
end for
return cost.
```

The time complexity of the algorithm is $O(100n)$.

Reference:

www.geeksforgeeks.com
www.cs.princeton.edu
www.cs.cornell.edu
people.cs.ksu.edu/~subbu/papers/minimumstops.pdf

Used these reference to learn about minimum fuelstop dynamic programming and job scheduling problems.