CMSC 641 Design & Analysis of Algorithms Spring 2023
Section 02 Homework 2

Faisal Rasheed Khan

VB02734

vb02734@umbc.edu

1 Greedy: Dam It

After graduation, you go work for the Army Corp of Engineers. They do civil engineering projects, but it's handy to have a computer science type on staff.

One day, your unit is sent to investigate a report of a leaky levee. At the site, your unit discovers that the levee has sprung n leaks. Water is spraying out of each leak at the rate of 10 gallon per minute. The civil engineer in your unit quickly determines that the ith leak will take t[i] minutes to patch up. Your unit can only patch one leak at a time, so it will take:

$$\sum_{i=0}^{n} t[i]$$

minutes to patch up all the leaks. However, as you are patching up the leaks, the unpatched leaks will continue to flood the area at the rate of 10 gallon per minute per unpatched leak. Which leak should you patch first?

For example, suppose n = 4, t[1] = 20, t[2] = 5, t[3] = 11 and t[4] = 8. If you fixed the patches in numerical order, leak #1 will stop leaking right away, leak #2 will leak for 20 minutes, leak #3 will leak for 20 + 5 = 25 minutes and leak #4 will leak for 20 + 5 + 11 = 36 minutes. A total of

$10 \cdot (20 + 25 + 36) = 10 \cdot 81 = 810$

gallons of water would have spilled. On the other hand, if you patched the leaks in a different order, say leak #4 first, followed by #2, #3 and #1), then only

$10 \cdot (8 + 13 + 24) = 10 \cdot 45 = 450$

gallons will be spilled.

Questions:

1. Describe the set of feasible solutions for a particular instance of this problem. What makes a solution feasible? What does the objective function measure?

A. The feasible solutions are patching up all the 'n' sprung leaks from the levee.

The objective function measures that the leakage of water should be minimized which is 10 gallon per minute.

2. Devise and describe a greedy algorithm for the problem above that minimizes the total amount of water spilled. State and briefly justify the running time of your algorithm.

A. Greedy Algorithm for Minimum water spilled:

- Given n leaks and $\sum_{i=1}^{n} t[i]$ be the i[th] leak that will take t[i] minutes to patch up.
- Sort the array t in increasing order, i.e., t[1]≤t[2] ≤*……≤t[n]*
- Iterate the t array from the front, let count=0
  Add the t[i] minutes, i.e., count=count[i-1], for i = 2 to n
  Assign t[i]=count, Repeat till n
- Return count*10

Greedy Algorithm for Minimum water spilled (Pseudo code):

```
count=0

// sorted array t in increasing order

for(i=2;i<=n;i++){

        count=count+t[i-1];

        t[i]=count;

}

return count*10;
```

The above greedy algorithm sorts the array and iterates through the array. Let the sorting algorithm used here is Merge sort.

O(n logn) + O(n)

O(n log n) for merge sort and O(n) for iterating the array of n elements

Running time complexity = O(n logn)

Space Complexity is O(n), for array t of n elements.

3. State a swapping lemma that can be used to prove that your greedy strategy produces the best solution. Note: Your swapping lemma should describe how to modify any feasible solution that is different from the greedy solution.

A. Suppose that Greedy Algorithm produces the sequence of water leaks, TimeA be its time to leak for a_i i=1 to n :

$A=(a_1, a_2, \ldots\ldots, a_n)$
$TimeA=(0, t[a_1], t[a_1]+t[a_2], \ldots\ldots\ldots\ldots, t[a_1]+t[a_2]+ \ldots\ldots + t[a_{n-1}])$
$\sum TimeA = A_{Total}$

Consider any other feasible solution, TimeX be its time to leak for x_i i=1 to n:

$X=(x_1, x_2, \ldots\ldots, x_n)$
$TimeX=(0, t[x_1], t[x_1]+t[x_2], \ldots\ldots\ldots\ldots, t[x_1]+t[x_2]+ \ldots\ldots + t[x_{n-1}])$
$\sum TimeX = X_{Total}$

Swapping Lemma,

$a_1 < x_1$, since Greedy Algorithm selects minimum time leak and a_1 conflicts with x_n in X.

Therefore, $TimeA_2$ ($t[a_1]$) < $TimeX_2$ ($t[x_1]$)

$t[a_1] - t[x_1] < 0$----------Eq(1)

let $a_1 = x_n$ (conflict element in X),

Then, we can replace x_1 in X with a_1 in A and x_1 in X with x_n in X and still have a feasible solution.

$X^{|} = (a_1, x_2, \ldots\ldots, x_1)$

$TimeX^{|} = (0, t[a_1], t[a_1]+t[x_2], \ldots\ldots\ldots\ldots, t[a_1]+t[x_2]+ \ldots\ldots + t[x_{n-1}])$

$\sum TimeX^{|} = X^{|}_{Total}$

Consider n^{th} element time of TimeX and $TimeX^{|}$,

$TimeX^{|}_n = t[a_1]+t[x_2]+ \ldots\ldots + t[x_{n-1}]$

$TimeX^{|}_n = t[a_1]+(TimeX_n - t[x_1])$

$TimeX^{|}_n = TimeA_2 +(TimeX_n - TimeX_2)$

From Eq(1), $TimeX^{|}_n < TimeX_n$ ----------Eq(2)

From Eq(2), $X^{|}_{Total} < X_{Total}$

Therefore, By Swapping Lemma Greedy Algorithm which is used for A produces the optimal solution rather than any feasible solution like X.

4. Argue that applying a single swap (as described in your swapping lemma) to a non-greedy feasible solution produces another feasible solution.

A. Suppose that Greedy Algorithm produces the sequence of water leaks, TimeA be its time to leak for a_i i=1 to n :

$A = (a_1, a_2, \ldots\ldots, a_n)$
$TimeA = (0, t[a_1], t[a_1]+t[a_2], \ldots\ldots\ldots\ldots, t[a_1]+t[a_2]+ \ldots\ldots + t[a_{n-1}])$
$\sum TimeA = A_{Total}$

Consider any other feasible solution, TimeX be its time to leak for x_i i=1 to n:

$X = (x_1, x_2, \ldots\ldots, x_n)$
$TimeX = (0, t[x_1], t[x_1]+t[x_2], \ldots\ldots\ldots\ldots, t[x_1]+t[x_2]+ \ldots\ldots + t[x_{n-1}])$
$\sum TimeX = X_{Total}$

Swapping Lemma,

$a_1 < x_1$, since Greedy Algorithm selects minimum time leak and a_1 conflicts with x_n in X.

Therefore, $TimeA_2$ ($t[a_1]$) < $TimeX_2$ ($t[x_1]$)

$t[a_1] - t[x_1] < 0$

let $a_1 = x_n$ (conflict element in X),

Then, we can replace x_1 in X with a_1 in A and x_1 in X with x_n in X and still have a feasible solution.

$X^{|} = (a_1, x_2, \ldots\ldots, x_1)$

$TimeX^{|} = (0, t[a_1], t[a_1]+t[x_2], \ldots\ldots\ldots\ldots, t[a_1]+t[x_2]+ \ldots\ldots + t[x_{n-1}])$

The sequence $X^{|}$ is still a feasible solution .

5. Argue that applying a single swap (as described in your swapping lemma) to a non-greedy feasible solution does not increase its objective value. (Some arithmetic is necessary here.)

A. Suppose that Greedy Algorithm produces the sequence of water leaks, TimeA be its time to leak for a_i i=1 to n :

$A = (a_1, a_2, \ldots\ldots, a_n)$
$TimeA = (0, t[a_1], t[a_1]+t[a_2], \ldots\ldots\ldots\ldots, t[a_1]+t[a_2]+ \ldots\ldots + t[a_{n-1}])$
$\sum TimeA = A_{Total}$

Consider any other feasible solution, TimeX be its time to leak for x_i i=1 to n:

$X = (x_1, x_2, \ldots\ldots, x_n)$
$TimeX = (0, t[x_1], t[x_1]+t[x_2], \ldots\ldots\ldots\ldots, t[x_1]+t[x_2]+ \ldots\ldots + t[x_{n-1}])$
$\sum TimeX = X_{Total}$

Swapping Lemma,

$a_1 < x_1$, since Greedy Algorithm selects minimum time leak and a_1 conflicts with x_2 in X.

Therefore, $TimeA_2$ $(t[a_1]) < TimeX_2 (t[x_1])$ and $a_1 = x_2$,

$t[a_1] - t[x_1] < 0$

let $a_1 = x_2$ (conflict element in X),

Then, we can replace x_1 in X with a_1 in A and x_1 in X with x_k in X and still have a feasible solution.

$X^{|} = (a_1, x_1, \ldots\ldots, x_k \ldots, x_n)$

$TimeX^{|} = (0, t[a_1], t[a_1]+t[x_1], \ldots\ldots\ldots\ldots, t[a_1]+t[x_1]+ \ldots\ldots + t[x_{n-1}])$

$\sum TimeX^{|} = X^{|}_{Total}$

Consider n^{th} element time of TimeX and $TimeX^{|}$,

$TimeX^{|}_n = t[a_1]+t[x_1]+ \ldots\ldots + t[x_{n-1}]$

$TimeX^{|}_n = TimeX_n$

$X^{|}_{Total}$ < X_{Total}

But still the elements can be interchanged and the time can be minimized.

$X^{||}_{Total} < X^{|}_{Total}$

$A_{Total} < X^{||}_{Total}$

So, swapping one element to non-greedy feasible solution does not increase its objective value.

6. Prove that the greedy algorithm does indeed produce a feasible solution with the optimum objective value. Note: You must show that it is not possible to patch the leaks in an order that results in less water spilled than from following your greedy algorithm. Do not appeal to any unproven general principles. You must incorporate your swapping lemma as you stated.

A. Proof of Induction on Minimum water spilled:

Induction Hypothesis P(n):

The greedy algorithm gives minimum water leak for n leaks.

Base Case:
P(1) holds since greedy algorithm repairs that leak and no water is spilled and that is optimum.

Induction Case:
Assume P(k) holds for all k<n. We want to show that P(n) holds.
Let S be some set of n leaks
Suppose that Greedy Algorithm produces the sequence of water leaks, TimeA be its time to leak for a_i i=1 to n :

$A=(a_1, a_2, \ldots\ldots, a_n)$
$TimeA=(0, t[a_1], t[a_1]+t[a_2], \ldots\ldots\ldots\ldots, t[a_1]+t[a_2]+ \ldots\ldots + t[a_{n-1}])$
$\sum TimeA = A_{Total}$

Consider any other feasible solution, TimeX be its time to leak for x_i i=1 to n:

$X=(x_1, x_2, \ldots\ldots, x_n)$
$TimeX=(0, t[x_1], t[x_1]+t[x_2], \ldots\ldots\ldots\ldots, t[x_1]+t[x_2]+ \ldots\ldots + t[x_{n-1}])$
$\sum TimeX = X_{Total}$

Swapping Lemma,

$a_1 < x_1$, since Greedy Algorithm selects minimum time leak and a_1 conflicts with x_n in X.

Therefore, $TimeA_2$ $(t[a_1]) < TimeX_2 (t[x_1])$

$t[a_1] - t[x_1] < 0$----------Eq(3)

let $a_1 = x_n$ (conflict element in X),

Then, we can replace x_1 in X with a_1 in A and x_1 in X with x_n in X and still have a feasible solution.

$X^{|}$ =(a_1 ,x_2 ,……,x_1)

Time$X^{|}$ = (0, t[a_1], t[a_1]+t[x_2], ……, t[a_1]+t[x_2]+ ….. + t[x_{n-1}])

$\sum TimeX^{|} = X^{|}_{Total}$

Consider n^{th} element time of TimeX and Time$X^{|}$,

Time$X^{|}_n$ = t[a_1]+t[x_2]+ ….. + t[x_{n-1}]

Time$X^{|}_n$ = t[a_1]+(TimeX_n – t[x_1])

Time$X^{|}_n$ = TimeA_2 +(TimeX_n – TimeX_2)

From Eq(3), Time$X^{|}_n$ <TimeX_n ----------Eq(4)

From Eq(4), $X^{|}_{Total}$ < X_{Total}

Recall S is the original set of n leaks

Let $S^{|}$ = S-{leak a_1 }

$S^{|}$ = S-{1}=n-1

On $S^{|}$ input, greedy algorithm gives,

$A^{|}$ =(a_2 , a_3,……,a_n)

But x_2 ,x_3 ,…., x_t do not conflict with a_1 . So,

$X^{|}$ =(x_2 ,x_3 ,……,x_t) is also a feasible solution for $S^{|}$,

By the Induction Hypothesis, we know that greedy minimizes the water leakage.

Thus,

Time$X^{|}_n$ = t[x_2]+ ….. + t[x_{n-1}]

Time$A^{|}_n$ = t[a_2]+ ….. + t[a_{n-1}]

Time$A^{|}_n$ < Time$X^{|}_n$

$A^{|}_{Total}$ < $X^{|}_{Total}$

which means A_{Total} < X_{Total}

Therefore, P(n) holds.

2 Cost of car ownership

In this question, we consider when it would be advantageous to sell the car you currently own and buy a new one. We assume that you have been given the following information from a "reliable" source:

- p(i) = the price of a new car in year i, for $1 \le i \le n$
- v(i, k) = the resale value of a car purchased in year i and sold in year k, for $1 \le i < k \le n+1$.
- m(i, k) = the maintenance cost during year k of a car purchased in year i, for $1 \le i \le k \le n$.

The problem is to determine the years y1, y2, . . . , yr when you would purchase new cars such that the total cost of car ownership from years 1 through n is minimized. The total cost is the sum of the maintenance costs for years 1 through n plus the price of each car purchased minus the resale value of each car sold. In addition, you should make the following assumptions.

- you don't have a car before year 1. (Since you have to buy a new car in year 1, y1 = 1.)
- you only want to purchase new cars.
- you are always able to sell your old car for v(i, k) dollars.
- at the end of n years, you sell your last car for v(yr, n + 1) dollars.

For example, if n = 10, y1 = 1, y2 = 5 and y3 = 7, then this solution states that you should purchase a new car in year 1, buy the second car in year 5 and buy the third car in year 7. (You would also sell the first car in year 5, the second car in year 7 and the third car at the beginning of year 11.)

Questions:

1. Define a function, MinCost, that can be used to solve this dynamic programming problem. To do this, you must describe the input parameters to MinCost and its "return value" using English sentences. Note: describe the value that MinCost must return in relation to its parameters, not how to compute it, that's the next question.

A. MinCost function contains 2 parameters i,j i.e. MinCost(i,j)
i iterates from 1 to n input,
j iterates from n-i+1 to n+1, for the car resale value

The return value would be:
$\text{Min}(\sum_{j=n-i+1}^{n+1}(\text{p}[\text{n}-\text{i}+1]+\text{m}(\text{n}-\text{i}+1,\text{j})\ -\ \text{v}(\text{n}-\text{i}-1,\text{j})\ +\ \text{MinCost}(\text{n}-\text{i},\text{j}))$

The base condition would be:

if(n==1)

MinCost(1,1) = (p[1] + m(1,1))

MinCost(1,2) = (p[1] + m(1,2))

…….

MinCost(1,j) =(p[1] + m(1,j))

2. Give a mathematical formula that shows how MinCost can be computed recursively. Then, explain the main parts of the formula in English.

A. MinCost(i,j) =

$$\sum_{i=1}^{n} Min \sum_{j=n-i+1}^{n+1} (p[n-i+1] + m(n-i+1,j) - v(n-i+1,j) + MinCost(n-i,j))$$

The base condition would be:

if(n==1) then

MinCost(1,1) = (p[1] + m(1,1))

MinCost(1,2) = (p[1] + m(1,2))

…….

MinCost(1,j) =(p[1] + m(1,j))

The sub problems starts from n and goes till 1(base condition). The base condition returns the computed value for one sub problem and therefore the value is computed from there on.

$MinCost(n-i, n-i+1)$ this function calls the remaining subproblems

$(p[n-i+1] + m(n-i+1,j) - v(n-i+1,j)$ computes the current car in ith year to be purchased and previous one to be sold.

3. Describe how MinCost can be computed bottom up using a dynamic programming table. Be sure to include a description of the dimensions of the table and the order that the entries of the table are to be filled. Which entry has the solution to the original problem?

A. MinCost(i,j) =

$$\sum_{i=2}^{n} Min \sum_{j=i+1}^{n+1} (\ p[i] + m(i,j)\ -\ v(i,j)\ +\ MinCost(i-1,j)\ )$$

The base condition would be:(p[1] + m(1,1),…..,p[1]+m(1,n)

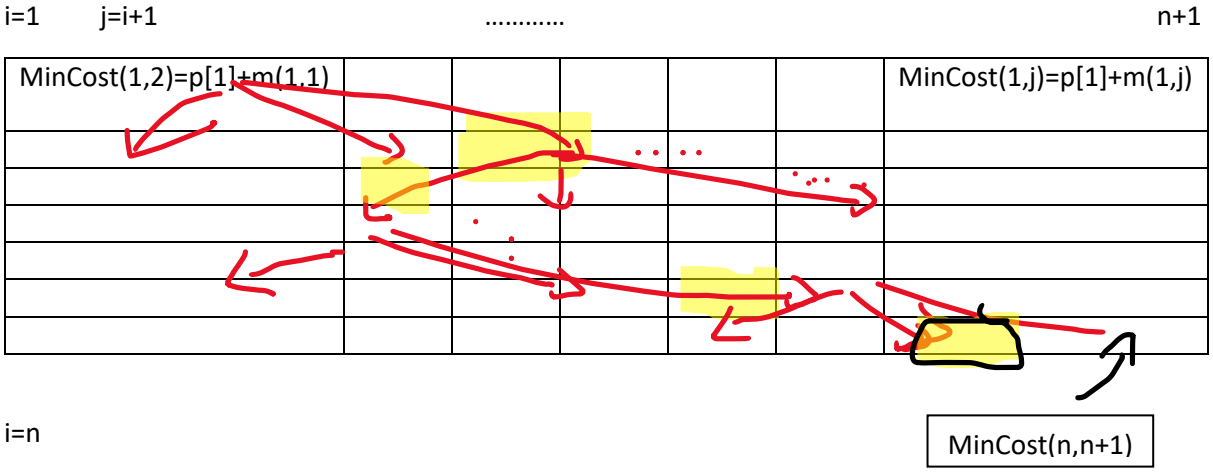
MinCost(1,1) = (p[1] + m(1,1))

MinCost(1,2) = (p[1] + m(1,2))

…….

MinCost(1,j) =(p[1] + m(1,j))

The MinCost function iterates from i=2, the initial base condition value of i is provided above.



First the initial base condition is computed for n=1;

With the initial base condition n=1, the remaiming subproblems are computed in bottom-up approach.

$$MinCost(1,1) = (p[1] + m(1,1))$$

$$MinCost(1,2) = (p[1] + m(1,2))$$

.....

$$MinCost(1,j) =(p[1] + m(1,j))$$

The next MinCost(2,j) values are computed and so on up to n.

The table has dimensions i x j for MinCost(i,j).

$1 \leq i \leq n$

$i < j \leq n+1$

n x n+1

MinCost(n,n+1) will have the answer for this bottom-up approach.

4. State and briefly justify the running time of your dynamic programming algorithm

A. MinCost(i,j) is the function
i for iterating the price p, j for iterating the resale value v from i+1
$1 \leq i \leq n$ ----> n iterations
$i < j \leq n+1$ ----> n-i+1 iterations = n iterations ($1 \leq i \leq n$)

The running time complexity = O (n * n)
The running time complexity = O (n^2)

References:

https://archive.org/details/introduction-to-algorithms-by-thomas-h.-cormen-charles-e.-leiserson-ronald.pdf/page/427/mode/2up

https://archive.org/details/AlgorithmDesign1stEditionByJonKleinbergAndEvaTardos2005PDF/page/n201/mode/2up

CMSC 641-01 Spring 2023 Shared Folder - Google Drive

CMSC 641-01 Spring 2023 Shared Folder - Google Drive

DAA Assign 1.docx