CMSC 641 Design & Analysis of Algorithms Spring 2023
Section 01 Homework 1 v1.1

Faisal Rasheed Khan

VB02734

vb02734@umbc.edu

Greedy Trick or Treat

Consider the following Trick-or-Treat problem. You live on a street with n houses. You have a bag to carry your Halloween candy, but your bag can carry at most K ounces of candy. You are given for each house i, an integer weight w_i (in ounces) of the candy that the people in house i are handing out. Each house gives out one piece of candy. You can visit each house at most once. Your problem is to collect the largest number of pieces of candy from the n houses without exceeding the capacity of your bag. Now consider the following greedy algorithm. Sort the houses according to the weight of the candy they are providing. Collect candy from the houses starting from the house that provides the lightest candy, then the next lightest, . . . until your bag is full.

Notes:

• The greedy algorithm for this problem has already been specified. You must prove that the given algorithm produces the optimum solution. Do not consider any other algorithms.

• Demonstrate that you are able to use a swapping lemma argument to prove the optimality of the greedy algorithm. Do not turn this into an algebra or calculus problem.

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 collecting candies with weight w_i from n houses without exceeding the capacity of the bag K, where visiting each house at most one time.

$\sum w_i$<=K; each house at most once

w_1+w_2 + ---- +w_i=W_{Total}

W_{Total} <=K

The objective function measures that there are as many as possible more number of candies whose weight w_i, when add up, should not exceed the capacity of the bag K, and visiting each house at most once.

2. State a swapping lemma that can be used to prove that this greedy strategy produces the optimum solution.

A. Suppose that Greedy Algorithm produces the sequence of collecting the candy weights from n houses, each house at most once:

A=(a_1 ,a_2 ,……,a_s) $\sum a_s$<=K; each house at most once

K – Capacity of the bag

a_1+a_2 + ---- +a_s=A_{Total}≤K

Consider any other feasible solution:

$X=(x_1, x_2, \ldots\ldots, x_t)$ $\quad \sum x_t <= K$; each house at most once

K – Capacity of the bag

$x_1+x_2+ \text{----} +x_t=X_{Total} \leq K$

Swapping Lemma,

$a_1 \leq x_1$, since Greedy Algorithm selects minimum candy weight from the house and a_1 does not conflict with X.

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

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

The total weight of X(X_{Total}) also changes, the new total weight of X($X^{|}_{Total}$)

$X^{|}_{Total}=X_{Total} -(x_1 - a_1)$

Since $a_1 \leq x_1$,

$X^{|}_{Total} \leq X_{Total}$; if $a_1 < x_1$, then there will be some available capacity in the bag for the candy with weight w_i to be included such that it should not exceed the capacity of the bag K.

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

3. 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 collecting the candy weights from n houses, each house at most once:

$A=(a_1, a_2, \ldots\ldots, a_s)$ $\quad \sum a_s <= K$; each house at most once

K – Capacity of the bag

$a_1+a_2+ \text{----} +a_s=A_{Total} \leq K$

Consider any other feasible solution:

$X=(x_1, x_2, \ldots\ldots, x_t)$ $\quad \sum x_t <= K$; each house at most once

K – Capacity of the bag

$x_1+x_2+ \text{----} +x_t=X_{Total} \leq K$ ----------Eq(1)

By Swapping Lemma,

$a_1 \leq x_1$, since Greedy Algorithm selects minimum candy weight from the house and a_1 does not conflict with X.

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

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

The total weight of X(X_{Total}) also changes, the new total weight of X($X^{|}_{Total}$)

$$X^{|}_{Total} = X_{Total} - (x_1 - a_1)$$

Since $a_1 \leq x_1$,

$$X^{|}_{Total} \leq X_{Total}$$ ----------Eq(2)

if $a_1 < x_1$, then there will be some available capacity in the bag for the candy with weight w_i to be included such that it should not exceed the capacity of the bag K.

From Eq(1) & Eq(2),

$X^{|}_{Total} \leq K$; which is a feasible solution

4. Argue that applying a single swap (as described in your swapping lemma) to a non-greedy feasible solution does not decrease its objective value.

A. Suppose that Greedy Algorithm produces the sequence of collecting the candy weights from n houses, each house at most once:

$A=(a_1, a_2, \ldots\ldots, a_s)$ $\sum a_s <= K$; each house at most once

K – Capacity of the bag

$a_1+a_2 + \text{----} + a_s = A_{Total} \leq K$

Consider any other feasible solution:

$X=(x_1, x_2, \ldots\ldots, x_t)$ $\sum x_t <= K$; each house at most once

K – Capacity of the bag

$x_1+x_2 + \text{----} + x_t = X_{Total} \leq K$

The total number of candy's before swapping in X are,

1, 2,……..,t=t candy's ----------Eq(3)

By Swapping Lemma,

$a_1 \leq x_1$, since Greedy Algorithm selects minimum candy weight from the house and a_1 does not conflict with X.

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

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

1, 2,……..,t=t candy's----------Eq(4)

From Eq(3) & Eq(4),

The total number of candies remains same i.e. 't'

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

5. Prove that the greedy algorithm produces a feasible solution with the optimum objective value.

A. Proof of Induction on Largest number of Candies:

Induction Hypothesis P(n):

The greedy algorithm selects more number of candies from n houses, with each house at most once, and with out exceeding bag capacity K.

Base Case:
P(1) holds since greedy algorithm will take the candy from the house if the candy weight w_1 will not exceed the bag weight K 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 candies taken from the n houses, each house at most once

Suppose that Greedy Algorithm produces the sequence of collecting the candy weights from n houses, each house at most once:

$A=(a_1, a_2, \ldots\ldots, a_s)$ $\sum a_s <= K$; each house at most once
K – Capacity of the bag

$a_1+a_2 + \text{----} + a_s = A_{Total} \leq K$
s candies

Consider any other feasible solution:

$X=(x_1, x_2, \ldots\ldots, x_t)$ $\sum x_t <= K$; each house at most once
K – Capacity of the bag

$x_1+x_2 + \text{----} + x_t = X_{Total} \leq K$----------Eq(5)
t candies

Swapping Lemma,

$a_1 \leq x_1$, since Greedy Algorithm selects minimum candy weight from the house and a_1 does not conflict with X.

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

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

The total weight of X(X_{Total}) also changes, the new total weight of X($X^{|}_{Total}$)

$X^{|}_{Total} = X_{Total} - (x_1 - a_1)$

Since $a_1 \leq x_1$,

$X^{|}_{Total} \leq X_{Total}$ ----------Eq(6)

From Eq(5) & Eq(6),

$X^{|}_{Total} \leq K$

if $a_1 < x_1$, then there will be some available capacity in the bag for the candy with weight w_i to be included such that it should not exceed the capacity of the bag K.

Recall S is the original set of candies taken from n houses, each house at most once.

Let $S^{|}$ = S-{candy a_1 from one house}

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

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

$$A^{|} =(a_2 , a_3,\ldots\ldots,a_s)$$

But $x_2 ,x_3 ,\ldots., x_t$ do not conflict with a_1 . So,

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

By the Induction Hypothesis, we know that greedy produces largest number of candies from n houses, each house at most once, without exceeding the bag capacity K with n-1 houses.

Thus,

Count($A^{|}$ candies) i.e. s-1$\geq$ Count($X^{|}$ candies) i.e. t-1

s-1 $\geq$ t-1

which means, s $\geq$ t

Count(A candies) i.e. s$\geq$ Count(X candies) i.e. t

Therefore, P(n) holds.

Elves at Helm's Deep

It is after the Battle of the Hornburg. The main characters have ridden off to Isengard, and you have been ordered to make sure Helm's Deep is secure. Since the Deeping Wall has been blown to bits, you decide to post guards further out at Helm's Dike instead. Helm's Dike is a long earthen wall that runs across full length of the valley of Deeping-coomb. The dike is quite steep and as tall as twenty feet in some places. However, you have found n places along the dike that may be vulnerable to attack. These are located x1, x2, . . . , xn yards from the western end of the dike. Your plan is to deploy elven archers along the dike. (Tolkien elves have keen eyesight, are excellent archers, don't sleep and are sort of immortal.) You figure that if you can place an elf within 50 yards of every vulnerable location then the dike will be well fortified.

Now, these elves are not actually under your command. They just came by to help out. Their leader, Haldir, met an unfortunate end during the battle, so they might accede to your request. In any case, you want to ask as few of them as possible to stand guard. (The whole idea of asking semi immortal beings to stand guard is somewhat awkward, but they don't need sleep!) Fortunately, you have taken algorithms and can find the minimum number of elven archers needed to accomplish this task.

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 deploying elves at vulnerable locations x_1 to x_n and atleast one elf is placed within 50 yards of every vulnerable location.

   The objective function measures that placing the number of elves at vulnerable locations should be minimized and there should be exact one elf within 50 yards that covers as much as many vulnerable locations.

2. Devise then describe a greedy algorithm that will find the locations to place the minimum number of elven archers along Helm's Dike so that each of the n vulnerable locations are within 50 yards of at least one elven archer.

A. <u>Greedy Algorithm Minimum Elf:</u>

- Given n vulnerable locations,
- x_1 , x_2 ,……,x_n in increasing order, let x be the array of the distances
- Assign the elf to the vulnerable location such that more number of vulnerable locations are covered by one elf within 50 yards.
- One elf should be placed at a vulnerable location such that it covers more number of vulnerable locations within 50 yards of the assigned elf vulnerable location.
- Let pos be the initial location of the n vulnerable locations, and elf be the count of the elves that are to be assigned to the vulnerable locations and cycle(0/1) variable ,initial 0, such that the elf covers right side of the locations when elf is assigned at the pos location.
- Iterate with the variable i from the initial position which is pos+1 until the last vulnerable location n.
- Compare the difference of the array of ith location from pos location, if it satisfies the condition of the difference within 50 yards then increment, else assign pos to i-1 and increment elf. cycle variable is assigned to 1 and the right side of the array is iterated where other vulnerable locations are covered from the point pos(elf assigned) until the difference of the array of ith location from pos location is not satisfied. After the condition of within 50 yards is not satisfied assign pos to i and complete the process.
- Repeat the above iteration until the last location and increment the elf accordingly such that the elves assigned to the vulnerable locations are minimum.
- x_1 , x_2 ,…..(Elf at x_f)……,x_k
- (Elf at x_f) covers from x_1 to x_k

n vulnerable locations
x_0 , x_1 , ……, x_{n-1} in increasing order
<u>Greedy Algorithm for Minimum elf (Pseudo Code):</u>

```
pos=0;

elf=0;

cycle=0;

if(n==0){
        return 0;
```

```
}
while(i<n){
    if(x[i]-x[pos]<50){
        if(x[i]-x[pos]<50 && i==n-1){
                elf=elf+1;          // Elf at position pos (increasing order of n within 50 yards)
        }
        i++;
    }
    else{
        if(cycle==1){
                pos=i;              //  second half of vulnerable locations within 50 yards
                cycle=0;
                elf=elf+1;
                                    // Elf at position i-1
                if(pos==n-1){
                        elf=elf+1;
                }
        }
        else{
                pos=i-1;            // first half of vulnerable locations within 50 yards
                cycle=1;
                i=i-1;
        }
        i++;
    }
  }
return elf;
```

The above greedy algorithm will find the locations to place the minimum number of elves. The running time complexity of the above greedy algorithm is O(n) where n is number of vulnerable locations. The space complexity is O(n), i.e. for array x of n elements.

Consider the scenario where elf can be place in between 50 yards which covers the entire x_1 to x_k within 50 yards. x_1 , x_2 ,…..(Elf at x_f)……,x_k . The 2nd else part in the algorithm checks till x_f and the other part of the valid locations will be checked by the 1st else-if part in the algorithm.

The position of the elf is determined by the difference of two vulnerable locations whose distance is not within 50 yards, and place the elf to the position-1 and check the other right side of the locations which meets the above difference criteria of exceeding 50 yards. Similarly, Then assign the new elf to the new position for the vulnerable locations with the above criteria till the end.

Consider 5 vulnerable locations at distance of 10,40,50,80,90

10	40	50	80	90

pos=0, i=1, elf=0

10	40	50	80	90

pos=0, i=2, elf=0

10	40	50	80	90

pos=0, i=3, elf=0

10	40	50	80	90

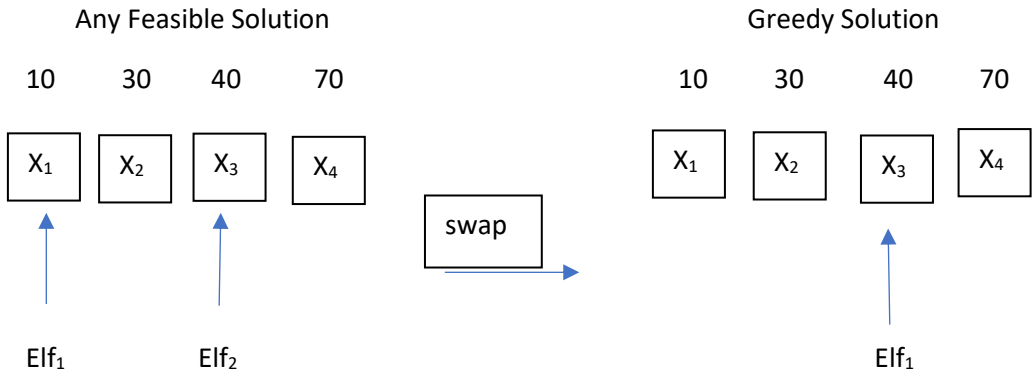
pos=2, i=3, elf=0

10	40	50	80	90

pos=2, i=4, elf=1(x_2)

3. State a swapping lemma that can be used to prove that this greedy strategy produces the optimum solution.

A.



minimum (Elf_1 + Elf_2, Elf_1)

Suppose that Greedy Algorithm produces the minimum number of elves for n vulnerable locations:

$A=(a_1, a_2, \ldots\ldots, a_s)$ $\quad \sum a_s$<n; n-number of vulnerable locations

a_1+a_2 + ---- $+a_s=A_{Total}$

Consider any other feasible solution:

$X=(x_1, x_2, \ldots\ldots, x_t)$ $\quad \sum x_t$<=n; n-number of vulnerable locations

x_1+x_2 + ---- $+x_t=X_{Total}$

Swapping Lemma,

Consider a_1 in A and x_1 , x_2 in X satisfies the condition of the distance between vulnerable location is within 50 yards

As greedy selects minimum elf,

Minimum(a_1 ,x_1 +x_2)=a_1 ;

Then, we can replace x_1, x_2 in X with a_1 in A and still have a feasible solution.

$X^{|}$ =(a_1 ,x_3 ,……,x_t)

The total of X(X_{Total}) also changes, the new total weight of X($X^{|}_{Total}$)

$X^{|}_{Total}$=X_{Total} -(x_1 +x_2 -a1)

$X^{|}_{Total}$=X_{Total} – 1

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

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

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 minimum number of elves for n vulnerable locations:

A=(a_1 ,a_2 ,……,a_s) $\sum a_s$<n; n-number of vulnerable locations

a_1+a_2 + ---- +a_s=A_{Total}

Consider any other feasible solution:

X=(x_1 ,x_2 ,……,x_t) $\sum x_t$<=n; n-number of vulnerable locations

x_1+x_2 + ---- +x_t=X_{Total}

Swapping Lemma,

Consider a_1 in A and x_1 in X satisfies the condition of the distance between vulnerable location is within 50 yards

As greedy selects minimum elf,

Minimum(a_1 ,x_1)=a_1 ; (either of a_1 or x_1)

Then, we can replace x_1, x_2 in X with a_1 in A and still have a feasible solution.

$X^{|}$ =(a_1 ,x_2 ,……,x_t)

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.

A. Suppose that Greedy Algorithm produces the minimum number of elves for n vulnerable locations:

$A=(a_1, a_2, \ldots\ldots, a_s)$ $\sum a_s$<n; n-number of vulnerable locations

$a_1+a_2 + \text{----} + a_s=A_{Total}$

Consider any other feasible solution:

$X=(x_1, x_2, \ldots\ldots, x_t)$ $\sum x_t$<=n; n-number of vulnerable locations

$x_1+x_2 + \text{----} + x_t=X_{Total}$

Swapping Lemma,

Consider a_1 in A and x_1 in X satisfies the condition of the distance between vulnerable location is within 50 yards

As greedy selects minimum elf,

Minimum(a_1 ,x_1)=a_1 ; (either of a_1 or x_1)

Then, we can replace x_1, x_2 in X with a_1 in A and still have a feasible solution.

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

The total of X(X_{Total}) also changes, the new total weight of X($X^{|}_{Total}$)

$X^{|}_{Total}=X_{Total}-(x_1-a_1)$

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

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

6. Prove that the greedy algorithm produces a feasible solution with the optimum objective value.

A. Proof of Induction on Minimum number of Elves:

Induction Hypothesis P(n):

The greedy algorithm assign minimum number of elves to n vulnerable locations, there should be exact one elf within 50 yards of every vulnerable location.

Base Case:
P(1) holds since greedy algorithm assigns the elf to that one vulnerable location 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 elves assigned to n vulnerable locations, exact one elf within 50 yards of every vulnerable location.

Suppose that Greedy Algorithm produces the sequence of assigning elves to n vulnerable locations, exact one elf within 50 yards of every vulnerable location:

$A=(a_1, a_2, \ldots\ldots, a_s)$ $\sum a_s < n$; n-number of vulnerable locations

$a_1+a_2+ \text{----} +a_s=A_{Total}$

s elves

Consider any other feasible solution:

$X=(x_1, x_2, \ldots\ldots, x_t)$ $\sum x_t < n$; n-number of vulnerable locations

$x_1+x_2+ \text{----} +x_t=X_{Total} \leq K$----------Eq(5)

t elves

Swapping Lemma,

Consider a_1 in A and x_1 , x_2 in X satisfies the condition of the distance between vulnerable location is within 50 yards

As greedy selects minimum elf,

$\text{Minimum}(a_1, x_1+x_2)=a_1$;

Then, we can replace x_1, x_2 in X with a_1 in A and still have a feasible solution.

$X^{|}=(a_1, x_3, \ldots\ldots, x_t)$

The total of $X(X_{Total})$ also changes, the new total weight of $X(X^{|}_{Total})$

$X^{|}_{Total}=X_{Total}-(x_1+x_2-a_1)$

$X^{|}_{Total}=X_{Total}-1$

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

Recall S is the original set of elves assigned to n vulnerable locations, exact one elf within 50 yards of every vulnerable location.

Let $S^{|}$ = S-{elf a_1 from one vulnerable location}

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

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

$A^{|}=(a_2, a_3, \ldots\ldots, a_s)$

But $x_2, x_3, \ldots., x_t$ do not conflict with a_1 . So,

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

By the Induction Hypothesis, we know that greedy assigns minimum number of elves to n vulnerable locations, exact one elf within 50 yards of every vulnerable location.

Thus,

Count($A^{|}$ elves) i.e. s-1≤ Count($X^{|}$ elves) i.e. t-1

s-1 ≤ t-1, which means, s ≤ t

Count(A elves) i.e. s≤ Count(X elves) i.e. t

Therefore, P(n) holds.

Code in C Language:

```
#include <stdio.h>


int main()
{
  int  pos=0;
int elf=0;int i=1;
int cycle=0;
//int n=6;int x[6]={10,40,50,110,130,140};
//int n=6;int x[6]={10,40,50,90,120,180};
//int n=10;int x[10]={10,40,50,90,120,130,140,150,190,195};
int n=8;int x[8]={10,40,50,90,145,150,190,195};
//int n=4;int x[4]={10,100,200,300};
//int n=4;int x[4]={10,20,30,40};
//int n=6;int x[6]={10,40,50,150,180,240};
/*if(n==0){
              printf("0 elf");
           //return 0;
}
*/
   printf("Here:\n");
   while(i<n){
      if(x[i]-x[pos]<50){
         if(x[i]-x[pos]<50 && i==n-1){
            elf=elf+1;
         }
```

```
            i++;
        }
        else{
            if(cycle==1){
                printf("%d%difloc\n",i,pos);
                pos=i;
                cycle=0;
                elf=elf+1;
                if(pos==n-1){
                    elf=elf+1;
                }
            }
            else{
                printf("%d%delseloc\n",i,pos);
                pos=i-1;
                cycle=1;

                i=i-1;
            }
            i++;
        }
        printf("%d%dglobloc\n",i,pos);
    }


    printf("%d",elf);

    return 0;
}
```

----------**Code Ends Here**----------

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