Chieh Lee
Problem set 4
You Zhang
Part I
1.

a.)
Let's assume penalty P = 5
the cost of each company for both weeks is the following

i	1	2
A[i]	20	10
B[i]	10	30
C[i]	12	12

with greedy algorithm approach, we choose company B for the first week, and then switch to company A since those two company has the cheapest price for each week plus the penalty is lower than the price. So we got 10+5+10=25
However, the optimal choice is to stick with company C which is 12+12=24
b.)
// create an array A for storing possible companies (a, b, c) for each week
array F[0…52][1…3]

F[0][1] = 0
F[0][2] = 0
F[0][3] = 0

// we can choose 3 companies for each week, and compare the minimum of three company cost of last week

For i = 1 to 52
F[i][1] = min(F[i-1][1], F[i-1][2]+P, F[i-1][3]+P) + A[i]
F[i][2] = min(F[i-1][1]+P, F[i-1][2], F[i-1][3]+P) + B[i]
F[i][3] = min(F[i-1][1]+P, F[i-1][2]+P, F[i-1][3]) + C[i]

// find out the path of how the optimal answer

For j = 52 to 1
If (i < 52)
F[i][select[i+1] = F[i][select][i+1]]-P

If min(F[i][1], F[i][2], F[i][3]) = F[i][1], select[i] = company A
If min(F[i][1], F[i][2], F[i][3]) = F[i][2], select[i] = company B
If min(F[i][1], F[i][2], F[i][3]) = F[i][3], select[i] = company C

c)
for given W weeks and C companies, we need to find the minimal cost among company (loop C) with backtracking to previous weeks (loop W).

2.
a)
String str = given string stored in 1 to str.length()
Array f[str.length]
f[0] = 0

// get max score

```
for i = 1 to str.length()
   f[i] = min
   for j = 0 to i - 1
      f[i] = max(f[i], f[j] + wordQuality(str[j+1...i]))
```

// backtracking the word
list l = new empty list
while str.length > 0

```
   maxScore = min
   for j = 1 to i
      if f[j-1] + wordQuality(str[j...i]) > maxScore
         maxScore = f[j-1] + wordQuality(str[j...i])
         maxWord = j
```

list.add(str[maxWord...i])
i = j-1

b)
because we have to consider all the possible words combination and the sum of the quality of words plus the optimal maximum score of the sentence. Therefore all the relevant segmentation is considered.
c)

assuming wordQuality is O(1), the worst case scenario is $O(L^2)$ where L is the length of string

3.

a)

Start vertex s, end vertex t

For each injured people, given a vertex p_i

For each hospital, given vertex q_i

No weighted edges

By using the max flow algorithm will guarantee optimal as many as injured people to go to hospital.

b)

for Ford-Fulkerson algo will be $O(N^2H)$

push-relabel methold will be $O(N^3H+NH^3)$

Ford-fulkerson algo seems more reasonable here.

4.

Part II

1.

the pseudocode for this doubly-linked list and its push operation:

push element **k** into the list **L**, list **L** is sorted

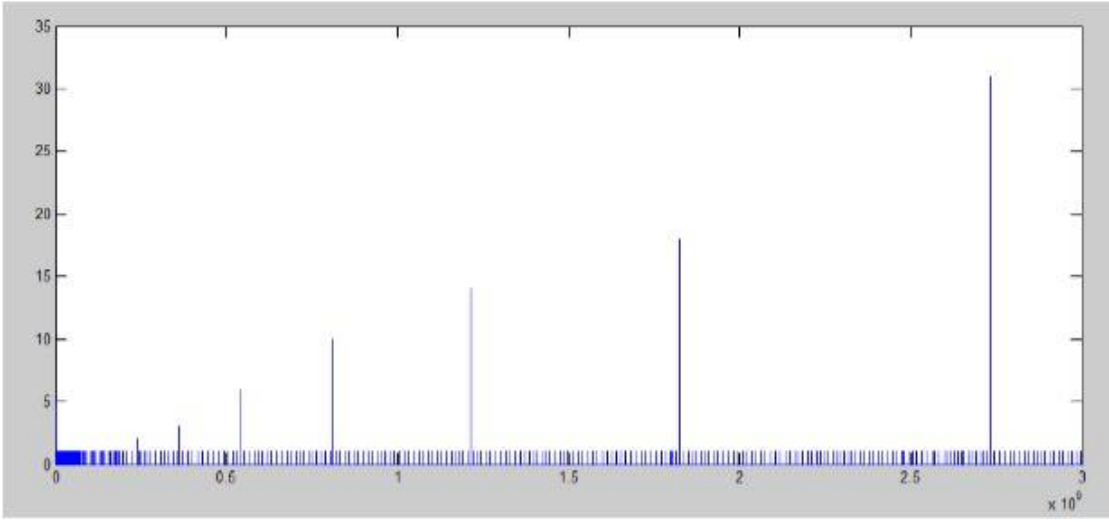
for **i** = 0 to the last element **n** in the list
if **k <= L[0] OR L** is **empty**
previous(L[0]) = k,
next(k) = L[0]
// L[0] become L[1], k become L[0]
return
else
previos(L[1]) = empty;
// L[1] become L[0], L[0] is dropped

i++

the worst case of this function is O(n), namely k is the biggest value of all. Operation will go through all elements compare with k and drop all elements.

Let the potential Φ(n) be the all elements in the list L. This starts 0 and is non-negative.
Push value k
The difference in potentials $\Phi(n_i) - \Phi(n_i\text{-}1)$

So when push element k,
the actual cost is **1**
the changing in Φ is going to be either or
(n + 1) – n = 1, amortized cost is 2 (when the element k is the re-point to the head of the list)
(n - 1) – n = -1, amortized cost is 0 (when element k is larger than head of list so the first element is dropped)
Since we charge 2 for each potential push, the worst case of the sequence of operation is **O(1)**



I think the spikes shows on the graph indicate ArrayList will grow its capacity when reach certain point. Because the capacity is growing larger and larger, the spike is looking larger as well.

4.
Programming part explanation

My code didn’t fully match the example from the problem set, instead the word is be edit from button to top (edit the latter char first). I could change what the example shows but I just ran out of time. I hope this isn’t considered bad achievement since it solves the problem as expected.