EE 602 – Algorithm I **(UH Manoa, Fall 2020)**

Algorithms Assignment 11

November 28, 2020

Instructor: Yingfei Dong *Name: Frendy Lio Can*

Problem 1

To prove the initial condition, we do the following: $n = 0 \Rightarrow T(0) = 2^0 = 1$.
For $n > 0, T(n) = 1 + \sum_{j=0}^{n-1} T(j) = 1 + \sum_{j=0}^{n-1} 2^j = 1 + (2^n - 1) = 2^n$

Problem 2

```
Algorithm(p,n,c)
  let r[n] be a new array with size n
  r[0] = 0
  for j = 1 to n
    q = p[j]
    for i = 1 to j - 1
      q = max(q, p[i] + r[j - i] - c)
    r[j] = q
  return r[n]
```

Problem 3

Let the vertices of the subproblem graph be the ordered pairs v_{ij} such that $i \leq j$. If $i = j$,then there are no edges out of v_{ij}. If $i < j$, then for every k such that $i \leq k \leq j$, the subproblem graph contain edges (v_{ij}, v_{ik}) and $(v_{ij}, v_{i+1,j})$.

Therefore, in order to solve the subproblem of optimally parenthesizing the product of A_i to A_j, we will need to solve subproblem of optimal parenthesizing the products A_i to A_k and A_{k+1} to A_j.

Number of vertices: $\sum_{i=1}^{n} \sum_{j=1}^{n} 1 = \frac{n(n+1)}{2}$

Number of edges:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(j-i) = \sum_{i=1}^{n}\sum_{a=0}^{n-i} t \quad \text{, a = j - i}$$
$$= \sum_{i=1}^{n}\frac{(n-i)(n-i+1)}{2}$$
$$= \frac{1}{2}\sum_{i=1}^{n}(n-i)(n-i+1)$$
$$= \frac{1}{2}\sum_{r=0}^{n-1}(r^2+r) \quad \text{,r = n - i}$$
$$= \frac{1}{2}(\frac{(n-1)n(2n-1)}{6} + \frac{(n-1)n}{2})$$
$$= \frac{(n-1)n(n+1)}{6}$$

Edges: (v_{ij}, v_{ik}) and $(v_{ij}, v_{i+1,j})$

Problem 4

Let a be an activity array.
Let c be an array that storages the sizes.
Let s be an array of the start time of the activities.
Let f be an array of the finish time of the activities.
Let n be the size of the original problem

```
DynamicActivitySelection(s,f,n)
  let $c$ and $a$ be a 2d array of size $(n+1)x(n+1)$
  for i in [0 ... n]
    c[i,i] = 0
    c[i,i+1] = 0
  c[n+1, n+1]= 0
  for l in [2 ... n + 1]
    for i in [0 ... n - l + 1]
      j = i + 1
      c[i,j] = 0
      k = j - 1
      while f[i] < f[k]
        if ( f[i] $\leq$ s[k] AND f[k] $\leq$ s[j] AND c[i,k] + c[k,j] + 1 > c[i.j] )
          c[i,j] = c[i,k] + c[k,j] + 1
          a[i,j] = k
        k = k - 1
  print "The maximum size set of mutually compatible activities has size:" c[0, n+1]
  print "The set contains:" PrintActivities(c, a, i, j)
```

```
PrintActivities(c, a, i, j)
  if[i, j] > 0
    k = a[i, j]
    print k
    PrintActivities(c, a, i, kj)
    PrintActivities(c, a, k, j)
```

Problem 5

If we choose the last activity to start that is compatible with all previously selected activities; the algorithm is similar to selecting the first activity to finish. The only difference it that when we do last activity to start, the algorithm will be running in reverse. Thus, since they are similar, it will produce an optimal solution.

This approach is a greedy algorithm because at each step we make the best choice.

Problem 6

Problem 7

For every stack operation, we add to our cost the actual cost of a stack operation and the cost of copying an element. This implies that for every stack operation, we add 2 times to the cost.

We also know that the size of stack will not exceed k, and there are always k operations between backups; we will overpay by at least enough cost. Therefore, the amortized cost of the operation is constant, and the cost of the n operation is $O(2n) = O(n)$

Problem 8

Using the INCREMENT Algorithm from page (454) from CLRS.

Let high be a global variable that is initialize at $-\infty$.

```
INCREMENT(A)
  i = 0
  while i < A.length and A[i] == 1
    A[i] = 0
    i = i + 1
  if i < A.length
    A[i] = 1
    if i > high
      high = 1
```

```
RESET(A)
  while high > 0
    A[high] = 0
    high = high - 1
  A[high] = 0
```

We can observe that INCREMENT(A) and RESET(A) operations will take $O(1)$ time by the accounting method. This is because when a bit is set to 1, we carry a cost of 1. This cost or bit can be used in the RESET operation as well which will take $O(1)$ time.

Therefore, the amortized cost is $O(n)$ which implies that the total cost is also $O(n)$.

Problem 9

If $\phi(D_i) = \sum_{k=1}^{i} lgk$, this implies that the amortized cost for INSERT is
$\hat{c_i} = c_i + \phi(D_i) - \phi(D_i - 1) \leq lgi + \sum_{k=1}^{i} lgk - \sum_{k=1}^{i-1} lgk = 2lgi \rightarrow O(lgn)$.

The amortized cost for EXTRACT-MIN will be
$\hat{c_i} = c_i + \phi(D_i) - \phi(D_i - 1) \leq lg1 + 1 + \sum_{k=1}^{i-1} lgk - \sum_{k=1}^{i} lgk = 1 \rightarrow O(1)$