CS6033 Assign No.5

Minghe Yang

March 2021

1 The partition problem

1.2 Of course it is a bad example. For a list of $[3, 4, 5, 300]$ in $k = 3$, the average will be 104, through this algorithm you'll get $[3, 4, 5]$, [300], and an empty list, while the right answer should be $[3, 4]$, [5], and [300].

1.3 For M(n, k), you have $n$elements with $(k - 1)$ separations. F n_{th} and $(n-1)_{th}$ element, what you need is to consider two situations. First: Compare $[t0]$ with $S(n-1, k-1)$. Second: Compare $[t0, t1]$with $S(n-1, k)$. If $n = k$ then everyone should be in a list, if $n < k$, return the biggest one since it's more likely to be an error.

1.4 The complexity of this algorithm is exponential,like $O(n^k)$ for all possibilities.

1.5 For some sub branches like those in FiboFibonacci, the one with same n, k, but different arrangements of the formal elements. like the same S(4,2) which refers to keep rest 4 elements in two lists.

1.6

Algorithm 1 DP Linear partition problem

Input: A given arrangement S of nonnegative numbers $[s1, ..., sn]$ and an integer k.

Output: Partition S into k ranges, so as to minimize the maximum sum over all the ranges.

```
function M(S, k)
    M[1, k] = s1
    M[n, 1] = \sum_{n=1}^{n} s_i
    M[n, k] = \min_{n=1}^{n} \max(M[i, k-1], \sum_{j=i+1}^{n} S_j)
end function
```

1.7 Let's start with the base case, from the definition we can surely know the $M[1, k]$ and $M[n, 1]$ is correct. As for $M[n, k] = \min_{n=1}^{n} \max(M[i, k-1], \sum_{j=i+1}^{n} S_j)$, it is the minimum of all the maximum cost of their sub Ms, so it is actually correct.

1.8 It is the number of cells times the running time per cell. A total of k $\cdot$ n cells exist in the table. For the worst case you need to compute j from 1 to n, so the total complexity will be $O(kn^2)$

1.9 You need to create another algorithm to test whether the $\max(M[i, k-1], \sum_{j=i+1}^{n} S_j)$ is equal to that cost, and print out every branches that is equal to the answer.

2 Critical thinking

2.

If n produce [0-4], then use double n rolls to produce 0-7.
$[0, 0], [0, 1], [0, 2]$ for number 0;
$[0, 3], [0, 4], [1, 0]$ for number 1;
$[1, 1], [1, 2], [1, 3]$ for number 2;
$[1, 4], [2, 0], [2, 1]$ for number 3;
$[2, 2], [2, 3], [2, 4]$ for number 4;
$[3, 0], [3, 1], [3, 2]$ for number 5;
$[3, 3], [3, 4], [4, 1]$ for number 6;
$[4, 1], [4, 2], [4, 3]$ for number 7;
If you get $[4, 4]$ finally, just reroll it. Currently there's no restriction for n, it doesn't even need to be positive, if you get a negative n, just append "-" for it.

3 Bellman-Ford algorithm

3.1

Algorithm 2 negative cycle detection

Input: A directed weighted graph $G =< V, E >$, two vertices s and t
Output: Check a negative cycle available or not

Do a regular Bellman-Ford Algorithm to judge all shortest paths for all vertices
Do another tmp judge for all vertices
if the shortest paths decrease for vertices **then**
    return True
elsereturn False
end if

4 Wifi Network

5

Algorithm 3 connection algorithm

Input: Location of k hotspots with range r and maximum load l, position of n clients

Output: Check whether all clients can connect to the Internet

List all cilents $[c_1, c_2, \ldots, c_n]$ and all hotspots $[h_1, h_2, \ldots, h_k]$ as vertices in two seperate rows
for Every hotspot $h_i \in [h_1, h_2, \ldots, h_k]$ **do**
  for Every client $c_j \in [c_1, c_2, \ldots, c_n]$ **do**
    if $distance[h_i, c_j] < r$ **then**
      Add edge $[hi, c_j, 1]$
    end if
  end for
end for
Add a vertex s that connect to all the clients with 1 capacity edges $(s- > c_i)$
Add a vertex t that connect to all the hotspots with maximum capacity l edges each $(h_i- > t)$
Apply Ford-Fulkerson algorithm to solve the maximum flow problem with the created graph
Make a cut including all h_i and t connection
if $capacityatminimumcut = n$ **then return** True
elsereturn False
end if

Proof: For every client, the input and output will be 1 if everyone is online. So the total flow will be$nclients * 1 = n$.Since every flow will be through that cut to get to t, so if the maximum flow is equal to n, then everyone's online.
The worst case for it will be $O(n*k)$(the situation where all clients are connected to every hotspot).