Sample Solution for Problem Set 13

Data Structures and Algorithms, Fall 2022

February 13, 2023

Contents

1 Problem 1

(a) Each vertex $v_{i,j}(1 \le i \le j \le n)$ represents an interval. So the number of vertices is

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

For each vertex $v_{i,j}$, its adjacent edges are $(v_{i,j}, v_{k,j}), i < k \le j$ and $(v_{i,j}, v_{i,k}), i \le k < j$, so the number of edges is

$$\sum_{i=1}^{n}\sum_{j=i}^{n} 2(j-i) = \frac{n^3-n}{3}$$

(b) In *MergeSort*, we divide a problem of solving (l, r) into (l, mid) and (mid, r). There are no overlapping subproblems in the whole process. (Any other reasonable argument is OK)

2 Problem 2

(a) Let f_n be the maximal income when cutting a rod with length n. Enumerate the length of the previous rod, we get

$$f_k = \max_{1 \le i \le k}\{f_{k-i} - c + p_i\}$$

Initially, $f_0 = 0$. Time complexity $O(n^2)$

(b) Let $f_{k,n}$ be the maximal income when cutting a rod with length n, into rods with length $\le k$. Enumerate the number of rods with length k, we get

$$f_{k,n} = \max_{0 \le i \le l_k \text{ and } i \times k \le n}\{f_{k-1,n-i\times k} + p_k \times i\}$$

Initially, $f_{0,0} = 0$. Time complexity $O(n \min\{\sum l_i, n \ln n\})$

3 Problem 3

Specify any node as the root of the tree, wlog, let v_1 be the root. Let f_i be the minimum size of vertex cover of subtree rooted at i when the vertex cover includes i. Similarly, let g_i be the minimum size of vertex cover of subtree rooted at i when the vertex cover does not include i. Initally, for any leaf v, $f_v = g_v = 0$. Then state transition is

$$f_u = 1 + \sum_{v \in son_u} \min\{g_v, f_v\}$$

$$g_u = \sum_{v \in son_u} f_v$$

We can do DFS from root, and compute f and g at the same time. Then answer is $\min\{f_1, g_1\}$. The total complexity is $O(n)$.

4 Problem 4

Let f_i be the contiguous subsequence of maximum sum ending at i. To be more specific, $f_i = \max_{1 \le j \le i}\{\sum_{t=j}^{i} a_j\}$. Initially, $f_0 = 0$. Then

$$f_i = \max\{0, f_{i-1}\} + a_i$$

The answer is $\max f_i, 1 \le i \le n$. Time complexity $O(n)$.
Note that we do not need an array to store f. Since we only use f_{i-1} to update f_i, we can use an variable to keep this value and make space complexity $O(1)$.

```
    ans = 0
    val = 0
    for i = 1 to n
        val = max(0, val) + a[i]
        ans = max(ans, val)
```

5 Problem 5

(a) $2, 10, 1, 1$

(b) Let $f_{i,j} (i \le j)$ be the maximal value the first player can get when playing this game at sequence $s_i, s_{i+1}, \cdots, s_j$. Then answer should be $f_{1,n}$. Note that for a subproblem $f_{l,r}$, if the first player choose to take s_i, then it comes to a subproblem of interval $[i+1, j]$ and the second player can get $f_{l+1,r}$. So the first player can get $\sum_{i=l+1}^{r} s_i - f_{l+1,r}$. Thus

$$f_{l,r} = \max\{\sum_{i=l}^{r} s_i - f(l+1, r), \sum_{i=l}^{r} s_i - f(l, r-1)\}$$

We can precompute $g_i = \sum_{i=1}^{n} s_i$ and compute $\sum_{i=l}^{r} s_i = g_r - g_{l-1}$. Thus we can do transition in $O(1)$. Total time complexity $O(n^2)$.

```
function solve():
    for i = 1 to n
        g[i] = g[i - 1] + s[i]
    for i = 1 to n
        f[i][i] = s[i]
    for len = 2 to n
        for l = 1 to n - len + 1
            r = l + len - 1
            f[l][r] = max(g[r] - g[l - 1] - f[l + 1][r]
                         ,g[r] - g[l - 1] - f[l][r - 1])
```

To recover the process the game or the optimal move of some player, the following process helps

```
function h(l, r, o):
    if f[l][r] == g[r] - g[l - 1] - f[l + 1][r] :
        player o choose s[l]
        h(l + 1, r, 1 - o)
    else
        player o choose s[r]
        h(l, r - 1, 1 - o)
```

6 Problem 6

Let $p_{i,j}$ be the probability that A will go on to win the match, when A has won i and B has won j. Initially, $p_{n,i} = 1, 0 \leq i < n$ and $p_{i,n} = 0, 0 \leq i < n$. When considering $p_{i,j}$, with probability $1/2$ A win next game and with probability $p_{i+1,j}$ A will go on to win the match. Similarly, with probability $1/2$ B win next game and with probability $p_{i,j+1}$ A will go on to win the match. By total probability formula, for $i, j < n$,

$$p_{i,j} = \frac{1}{2} \times p_{i+1,j} + \frac{1}{2} \times p_{i,j+1}$$

then we can compute all $p_{i,j}$ in a descending order of i, j in $O(n^2)$.

7 Problem 7

Let f_i be the minimum slop of the first i words of the paragraph. To compute f_i, we only need to enumerate the number of words in the last line, and compute the cost. See the following for details.

```
for i = 1 to n
  l[i] += l[i - 1]
for i = 1 to n
  f[i] = inf
f[0] = 0
for i = 1 to n
  for j = 1 to i
    v = L - (i - j) - (l[i] - l[j - 1]);
    if (v >= 0)
        if (i == n)
          f[i] = min(f[i], f[j - 1]);
        else
          f[i] = min(f[i], f[j - 1] + v * v * v);
```