181240045 潘蕴之 Problem Set 9

Problem 1: 1° If all edge weights $\in [1, |V|]$, we can use BucketSort or other linear-time sort algorithms, which costs $\Theta(|E|+|V|)$ time. Then we do the following thing just as Kruskal's algorithm. And the time complexity is $\Theta(|E|+|V|) + O(|V|) + O(|E|\log^*|V|)$ $= O(|E|\log^*|V|)$, since $|V| = O(|E|)$ for a connected tree.

2° The algorithm is the same as above, so the total time is 10

$\Theta(|E|+W) + O(|V|) + O(|E|\log^*|V|) = O(W + |E|\log^*|V|)$

Problem 2. (a) 不妨设 $e=(u,v)$，由于 $e \notin E'$，不妨考虑 Kruskal's algorithm 所以在 e 被选中时，u, v 必属于同一 cc，e 不可能加入到 MST 中。

又由于 $\hat{w}(e) > w(e)$，modify 后的 e 必在 modify 前 e 之后被选中 ⇒ 仍然不会加入到 MST 中 ⇒ in this case, we need to do nothing.

(b) ~~Suppose e=(u,v)~~ 12

UpdateMST:

$T' = T \cup \{e\}$

In T', use BFS to find a cycle and record the weight of each edge in this cycle.

Suppose the heaviest edge in the cycle is f

return $T' - f$

(c) we need to do nothing

(d) UpdateMST:

$T' = T - e$

T' consists of two CCs, suppose the vertex set of one CC is S, then that of the other CC is $V-S$.

Minedge = e

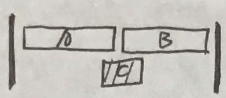
for each $(u,v) \in E$:

if (u,v) crosses the cut $(S, V-S)$ and $(u,v).w <$ Minedge.w,

Minedge = (u,v)

Problem 3:

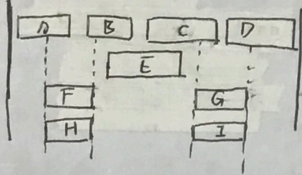
Least duration:



There are three activities A, B, C and their start time and finish time are shown left. Using this approach, we will choose C. However, the optimal solution is {A, B}.

10

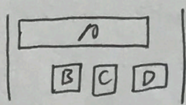
Fewest overlap:



This approach: {A, E, D}

optimal solution: {A, B, C, D}

Earliest start time:



This approach: {A}

Optimal solution: {B, C, D}

Problem 4:

Algorithm: Keep taking the most valuable item until the knapsack is full.

Proof: ① greedy-choice: let a_m be a most valuable item that can fit into the bag, then in some optimal solution, this item is taken.

- Consider an optimal solution, assume a_m is not taken.
- Since a_m is the most valuable item, it must have the least weight. We can always substitute another item of weight $w' > w_m$ in the bag with a_m.
- The new solution cannot be worse since a_m is the most valuable.

② optimal substructure:

let a_m be the most valuable item in the item set S. Then "$OPT(S-a_m)$" is an optimal solution of the problem.

- Considering some $OPT(S)$ containing a_m.
- If optimal substructure does not hold, then $OPT(S)$ gives $SOL(S-a_m) > OPT(S-a_m)$, which contradicts the optimality of $OPT(S-a_m)$.

Problem 5: 假设all 256 characters 为 $\{c_1, c_2, \cdots, c_{256}\}$，其对应的 frequency 为 $\{f_1, f_2, \cdots, f_{256}\}$, where $f_1 \le f_2 \le \cdots \le f_{256}$, $2f_1 > f_{256}$.

The total length using 8-bit fixed-length code is:

$$\sum_{i=1}^{256} 8f_i = 8\sum_{i=1}^{256} f_i$$

考虑 Huffman coding 每一步最小的两个 frequency:

step 1: $f_1 \le f_2 \le \cdots \le f_{256} \Rightarrow f_1, f_2 \Rightarrow$ new node is f_1+f_2

10

step 2: 由于 $2f_1 > f_{256}$，有 $f_1+f_2 > f_{256}$

$\Rightarrow f_3 \le f_4 \le \cdots \le f_{256} \le f_1+f_2 \Rightarrow f_3, f_4 \Rightarrow$ new node is f_3+f_4

step 3: 显然有，$f_3+f_4 > f_{256}$，$f_3+f_4 \ge f_1+f_2$

$\Rightarrow f_5 \le f_6 \le \cdots \le f_{256} \le f_1+f_2 \le f_3+f_4$

同理，

⋮

step 128: new node is $f_{255}+f_{256}$

$\Rightarrow f_1+f_2 \le f_3+f_4 \le \cdots \le f_{255}+f_{256}$

step 129: $f_5+f_6 \le \cdots \le f_{255}+f_{256} \le \underbrace{f_1+f_2+f_3+f_4}_{\text{new node!}}$

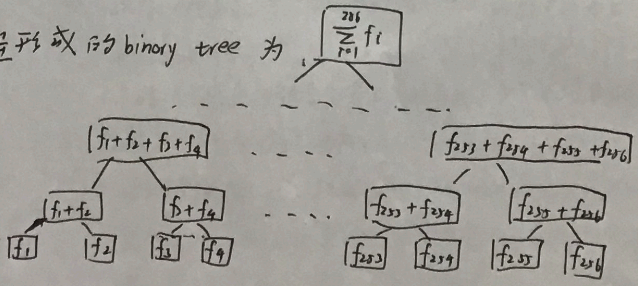
$\because$ $f_{256} < f_1+f_2$, $f_{256} \ge f_{255}$, $f_1+f_2 \le f_3+f_4$

⋮

step (128+64) 192: $f_1+f_2+f_3+f_4 \le f_5+f_6+f_7+f_8 \le \cdots \le f_{253}+f_{254}+f_{255}+f_{256}$

⋮

所以上述过程形成的 binary tree 为



$\Rightarrow$ 每个 leaf node 的 depth 均为 $\lg 256 = 8$

$\Rightarrow$ total length is also $8\sum_{i=1}^{256} f_i$, which is no more efficient than ordinary 8-bit fixed-length code.

Problem 6: 6

Algorithm: GreedyColor.

- Suppose each color is represented by a positive integer, then the color set $C = \{1, 2, 3, \cdots\}$
- Construct interval set S, where $S[i] = (L[i], R[i])$, $1 \le i \le n$
- Construct a set U, where $U[i]$ ~~is~~, $1 \le i \le n$. is the color of $S[i]$.

```
for i=1 to n:
    U[i] = 0   // initialize
max_color = 0
sort S according to the start time L[i] by increasing order.
for i=1 to n:
    O = ϕ
    for j=1 to i-1:
        if S[i] overlaps S[j],
            add U[j] to O
    U[i] = min{l | l ∈ C-O}
    max_color = max(max_color, U[i])
return max_color
```

Time complexity: $O(n^2)$ 太慢]

Correctness: [Greedy choice]: 假设~~当前已经涂色的interval的颜色集合为U'~~。~~集合为S，对应的~~
等待涂色的当前interval为s，从颜色全集C中除去与s overlap的interval的颜色后，其中的最小值即为s的颜色。

证：
- 设与s冲突的interval的颜色集为c'，$l = \min\{c \mid c \in C-C'\}$，$k > l$且$k \in C-C'$.
- 不妨假设s的颜色不是l，而是k。于是已涂色的interval的颜色集$C'' = C' \cup \{k\}$，$OPT(C'') = \max\{c \mid c \in C''\}$.
- 将k替换为l后，'any two overlapping intervals are assigned different colors'这条性质并不会被破坏。并且由于$l < k$，$OPT(C'')$不可能变得更大。即得证。

[Optimal substructure]: ~~设已经涂色的interval的颜色集合设~~ 设当前interval的正确上色为l，M为所有interval正确颜色的集合，则

$\max(OPT(M-\{l\}), l)$ 为后问题的最优解。

Proof: 假设 $\max(OPT(M-\{l\}, l))$ 不是后问题最优解

$\Rightarrow$ ~~max(max(M-{l}), l) 共 max{~~

$\max(\max\{c \mid c \in M-\{l\}\}, l) \neq \max\{c \mid c \in M\}$

$\Rightarrow$ 矛盾.

Problem 7:

Algorithm:

```
MaxClimber(T, k)
    paths = 0
    Start from the root to compute the depth of all nodes in T and sort them decreasingly. ---- ①
    while begin = find_the_deepest_node_in_T() : ---- ②
        moves = 0
        while moves < k and begin.parent != NIL
            begin = begin.parent
            moves = moves + 1
        if moves == k : // success find a path
            paths = paths + 1
            Delete the subtree rooted at begin.
        else : // there must be no path!!
            break
    return paths
```

这一步怎么O(1)做的？不是删除了很多结点吗？

这一步怎么做的？

Time complexity:

Suppose there are n holds totally. step ① takes: $O(n\lg n)$

② : while loop will progress $O(\frac{n}{k}) = O(n)$ times. each time takes $O(1)$

$\Rightarrow$ total time is $O(n\lg n) + O(n) = O(n\lg n)$

Correctness: [Greedy choice]: 假设 a_m 是当前 deepest 的节点，其 k-edges-path 的终点是 a_n. 记 $a_m \rightarrow a_n$ 的唯一路径为 p，则 p 必在 optimal solution 中.

proof: 假设 p 不在 optimal solution 中。我们可以将 p 添加到 solution 中，则此时必会产生 conflict (否则 optimality 不满足)，记与 p 'touch the same hold'/'intersect'

的路径为q。

·从 optimal solution中删去q，conflict消失了，同时原本q的edge仍可以形成其他的可行path
$\Rightarrow$ the new solution cannot be worse.

[optimal substructure].

设a_m为deepest节点，其k-edges-path终点为a_n，路径$a_m \to a_n$为p，以a_n为根的subtree为T'，则$OPT(T-T') \cup \{p\}$是问题的optimal solution.

proof：首先证明T'中除p外，必不存在k-edges-path.

显然，T'中的最长路径为k，并且一端为a_n—T'的根节点。由性质'no two climbers are allowed to touch the same hold' $\Rightarrow$ 除p外，其余path的长度必小于k。

假设$OPT(T-T') \cup \{p\}$不是optimal solution

$\Rightarrow$ $T-T'$非optimal substructure

$\Rightarrow \exists \left|SOL(T-T')\right| > |OPT(T-T')|$，与$OPT(T-T')$的optimality矛盾。