Algorithmn HW8

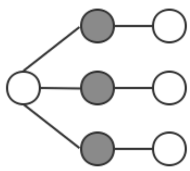
5140379032 JIN YI FAN

Problem 8.5

Require: total value of money to pay $n < 2^{k+1}$
Ensure: combination plan $plan[0 \dots k] = \{c_0 \dots c_k\}$, c_i refers to number of coin values 2^i

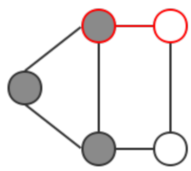
$plan[0 \dots k] \leftarrow 0$
for $i \leftarrow k$ to 0 **do**
    $plan[i] \leftarrow \lfloor n/(2^i) \rfloor$
    $n \leftarrow (n - plan[i] * 2^k)$
end for
return $plan$

Problem 8.7



If we take the introduced greedy approach, we may get the vertex cover as all the 4 white vertexes. But actually it should be the set of 3 grey vertexes.

Problem 8.8



If we take the introduced greedy approach, very likely we will delete the left single vertex and get the red clique, but actually the grey clique is the required one.

Problem 8.27

Yes.
Because actually we can add a big enough constant to all the weights of edges to make sure each weight is a positive number. MST of this graph won't change.

Problem 8.28

Assumptions: T_1, T_2 are two minimum spanning trees of $G(V, E)$, and $T_1.E = (a_1, a_2 \dots a_k \dots a_n)$, $T_2.E = (b_1, b_2 \dots b_k \dots b_n)$, both ordered by weight in ascending order and k is the minimum subscript s.t. $weight(a_k) \neq weight(b_k)$, thus $a_k \notin T_2.E$ and $b_k \notin T_1.E$.
We can assume that $weight(a_k) < weight(b_k)$ in this case.
Then $T_2 \cup a_k$ must contain a cycle C.
$\because \forall i < k, a_i = b_i$, and $\{a_1 \dots a_{k-1}, a_k\}$ shouldn't become cycle,
$\therefore \{b_1 \dots b_{k-1}, a_k\}$ won't become a cycle,
$\therefore \exists b_j \in C,\ s.t.\ weight(b_j) > weight(a_k)$,
$\therefore T_2$ shouldn't be a minimum spanning tree of G, which leads to a contradiction.

Maximum Spanning Tree

Require: A weighted graph $G = (V, E)$

Ensure: The maximum spanning tree $T = (V_M, E_M)$

$sum \leftarrow \sum_{e_i \in E} weight(e_i)$
for each $e \in E$ **do** ▷ reverse the weights of the edges in E
    append $(u, v, (sum - weight(e)))$ to E_r
end for
$(V_M, E_M) \leftarrow \textsc{Prim}(V, E_r)$ ▷ do Prim
for each $e \in E_M$ **do** ▷ recover the weights of the edges in E_M
    $weight(e) \leftarrow (sum - weight(e))$
end for
return (V_M, E_M)