HomeWork4

1827406008 厉张子健

November 2020

1 第一题

假设 $G=(V,E)$ 存在两个最小生成树 $T_1(e_1,e_2,e_3...e_m)$, $T_2(e_1^{'},e_2^{'},e_3^{'}...e_m^{'})$ 从 T_1 中取出一个 e_k, 在 T_2 中加入该边构成环, 由于各边权值不同, 其中必然存在一个 e_k 权值大于 $e_j^{'}$, 而根据已知条件可知这个树不是最小生成树, 与之矛盾.

所以只能有唯一一个最小生成树.

2 第二题

算法简述: 对于一个边 (e,v), 从一个节点 e 出发进行 dfs, 搜索节点 v, 过程中遇到的权重比这个边大的跳过, 如果能够搜索到并且不是通过 (e,v), 证明这两个节点在同一个环上, 而环上别的边的权重都比 (e,v) 小, 根据红色规则, 这个边标记为红色, 不在最小生成树中; 如果不能搜索到, 证明在最小生成树上。DFS(e): 将所有节点初始化为未检索

设置一个栈 S, 其中加入节点 e

while S 非空

 pop 出 S 中的栈顶节点 h

 if h 没有被扫描过

  标记 h 为已扫描

  对于 h 的邻接表中的任意边 $m(h,j)$

   if $c_m > c_e$

    continue

   if $j == v\ and\ e! = h$

返回错误, 结束循环

push j 到 S 中

返回正确

3 第三题

因为一个基站的覆盖范围只有 $(\varepsilon-4, \varepsilon+4)$, 为了尽可能的减少基站数目, 但是要让所有房屋都在基站范围内, 所以从左到右, 如果 8 公里的范围内有房屋, 将最左侧房屋放置到 $\varepsilon-4$ 位置, 中间放置基站, 在这个基站范围内的房屋都被覆盖, 当有房屋位置超出 $\varepsilon+4$ 时，如果这个房屋到 $\varepsilon+4$ 之间没有房屋，这个房屋就是剩余房屋的最左侧房屋，重复上述操作。这样所有房屋都在基站范围内。

设最左边的节点坐标为 0

Arrange(int[] house):

设置列表 list 记录基站 (station) 位置

设置第一个基站位置为 4, 并加入列表

设置变量 x 记录基站位置, 初值为 4

$for(int\ i=0; i<$ 房屋数目$;i++)$:

  if($house[i] > x$):

    $x = house[i] + 4$

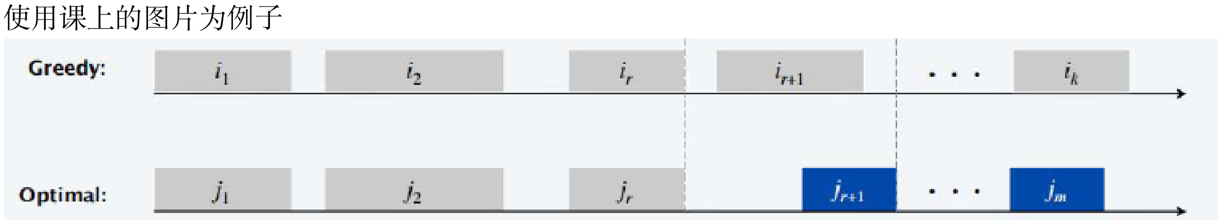
    在 list 中加入 x 的新值

return list

设房屋数目为 n，如果房屋列表无序，那么需要先排序，采用归并排序，时间为 $O(n\log_2 n)$, 后面设置基站, 时间复杂度为 $O(n)$, 所以总体的时间复杂度为 $O(n\log_2 n)$; 如果房屋有序, 则为 $O(n)$

正确性：

使用课上的图片为例子



加入存在一个更好的算法，那么第 $r+1$ 个基站就不会以第一个未覆盖

的房子, 那么在 j_r 和 j_{r+1}) 之间就存在未被覆盖的房子, 需要增加一个基站, 因此该算法需要的基站数目更多。因此贪心算法是最优的，同时也可以保证所有的房屋都在基站覆盖范围内。

4 第四题

是正确的:

证明如下:

记第 k 辆车携带的货物质量为 T_k, 剩余货物总重量为 W, 总共用车 n 辆

假设存在一个更优解:

设更优解与贪心算发在第 k 辆车开始出现区别, 那么 T_k 和 $T_k^{'}$ 不相等, 其中 $k \geq 1$

在贪心算法中, 每辆车都是尽可能装入货物, 而新算法不是采取这样的策略, 那么 T_k 一定大于 $T_k^{'}$

那么相比于贪心算法, 更优解剩余货物至少为 $W \cup w_i$

那么剩余的货物至少多出 1 件, 质量多出 w_i

如果更优解 k 之后的所有车的 $T_j^{'}$ 都和贪心法一样, 那么需要多安排一辆车, 显然矛盾

如果多出的货物尽可能放入其他车, 那么这个更优解就是贪心法调整顺序后的解法

同时, 由于更优解使用的车数目更少, 减少的 m 辆车中的货物需要放入其他 $n-m$ 辆车中, 说明贪心算法并没有尽可能装入货物, 矛盾

因此, 贪心算法是最优解

5 第五题

采取如下的贪心算法:

定义 $a_i = w_i/t_i$ 表示一个任务的平均权重:

将任务按照 a_i 逆序排序

优先执行 a_i 高的任务, 按照顺序执行即可以

首先证明引理，当所有任务消耗时间相等时，权重更大的优先执行所求加权和更小:

假设耗时都是 t, 第 i 个任务的权重为 w_i
假设 n 个任务已经按照最小顺序安排好了，加权和为 T, 现在插入一个权重比 n 个任务中最大的还要大的任务。
如果将任务放到第 k 个位置，后面所有任务的 c_i 需要加 t, 所以, 加权和为

$$M_k = kw_{n+1}t + \sum_{i=1}^{k-1} w_i c_i + \sum_{i=k+1}^{n+1} w_i(c_i + t) = kw_{n+1}t + T + \sum_{i==k+1}^{n+1} w_i t$$

用 $k+1$ 的加权和减去 k 的加权和, 得到

$$M_{k+1} - M_k = w_{n+1}t - w_{k+1}t$$

而 $w_{n+1} > w_{k+1}$, 所以 k 越小, 权重和越小。故权重更大的优先执行时所求加权和更小。

下面证明这个算法的正确性:
a_i 可以理解为每一分钟任务所占的权重，那么根据引理可知，我们需要优先执行 a_i 大的那一分钟任务，对于剩余的任务，继续这样的排序。由于同一个任务的每一分钟的权重相等，于是按照每一分钟的权重排序后，同一任务的不同分钟会排在相邻位置，这样也完成了任务的排序。

6 第六题

运行结果和时间如下:

```
Prim求得大图MST权重和-3612829
Prim大图用时361
Kruskal求得大图MST权重和-3612829
kruskal大图用时32
Prim求得小图MST权重和14
Prim小图用时1
Kruskal求得小图MST权重和14
kruskal小图用时1
```

可以看出 kruskal 算法更快