补充题解 - 《经典》- 第 8 章高效算法设计

习题 8-3　比特变换器（ Bits Equalizer, SWERC 2012, UVa12545）

首先要忽略S和T中已经相同的位置。分别记录以下4种情况出现的次数:

1. S[i] = 0, T[i] = 1, 记为s01.
2. S[i] = 1, T[i] = 0, 记为s10.
3. S[i] = ?, T[i] = '0', 记为q0.
4. S[i] = ?, T[i] = '1', 记为q1.

记所求结果为ans = 0, 首先尽量将s01和s10中的位置进行互换，记x = min(s01, s10)，则:

```
ans = x + q0, s01 -= x, s10 -= x
```

此时如果s10 > q1，参考如下的情况，因为此时只能由'?'变成0，于是就无法产生足够的'0'，返回-1即可。

```
1111????
00001100
```

否则可以先把?变成0，然后再和s10中的1进行交换即可。

最后，ans += s10+s01+q1。具体含义就是如下操作的次数之和，参考如下的情况:

```
11????
001100
或者
00????
111100
```

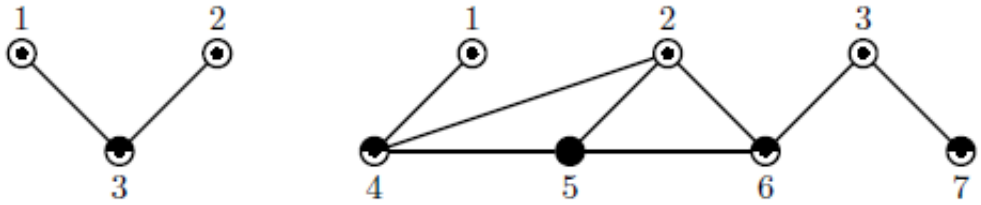
- 0->1
- ?->0 之后和 1->0 交换

- ?->1

习题 8-9 Graph Oddity, ACM/ICPC NEERC 2010, UVa1613

输入一个n（ 3 ≤ n ≤ 9999）个 点 m条边（ 2 ≤ m ≤ 100000）的连通 图， n 保证为奇数。 设k为最小的奇数， 使得每个点的度数不超过k， 你的任务是把图中的结点涂上颜色1～k， 使得相邻结点的颜色不同。 多解时输出任意解。 输入保证有解。

如下图所示， k=3。



【分析】

首先建图的时候就可以顺路求出k。然后不难想到使用BFS，每次遍历到一个结点u，看看K个颜色中，有哪些已被与u相邻的结点使用。在剩下的颜色中选择一个使用即可。

下面我们考虑证明这个做法的正确性。

记最大度数结点为u，u的度数为D。如果D为偶数，那么显然k种颜色都是足够的。

如果D为奇数，并且和u相邻的点的颜色各不相同，只有和D连接的所有结点v，都互相连接形成完全图的情况下，k种颜色才不够用，此时每个v的度数都已经为D。那么不能再连接除了这个完全图中的其它结点，所以u和其相邻的所有v就形成了整张连通图，图的结点个数就是D+1为偶数，与n为奇数矛盾。所以k种颜色一定是够用的。

习题8-10 奇怪的股市(Hell on the Markets,ACM/ICPC NEERC 2008, UVa1614)

输入一个长度为n（n<=100000）的序列a，满足1<=ai<=i，要求确定每个数的正负号使得所有数的总和为0。比如a={1, 2, 3, 4}，则设四个数的符号分别是1, -1, -1, 1即可（1-2-3+4=0），但如果a={1, 2, 3, 3}，则无解（输出No）。

【分析】

本质上是要把序列划分成两组，两组的绝对值之和相等，然后分别标记成不同的正负号即可。记 $S=\sum_{i=1}^{n} a_i$。则显然当S为奇数时，问题无解。

按照i从大到小的顺序访问a_i，取出部分元素作为一组。记录当前已取出部分之和为T，如果 $2\cdot(T+a_i)\le S$则取出a_i，否则不取。

下面证明上述算法的正确性：考虑最靠左的没被取的a_i以及判断a_i之前的T，记此时的T值为Q。那么有：

$$Q+a_i>\frac{S}{2}\implies i\ge a_i>\frac{S}{2}-Q\implies Q+i-1\ge\frac{S}{2}$$

而且根据题意有：

$$a_i \geq 1 \Longrightarrow Q + \sum_{k=1}^{i-1} a_k \geq Q + i - 1 \geq \frac{S}{2}$$

根据算法描述有:

$$Q + \sum_{k=1}^{i-1} a_k \leq \frac{S}{2}$$

所以有: $Q + \sum_{k=1}^{i-1} a_k = \frac{S}{2}$，也就是说最终取到的数字之和一定为$\frac{S}{2}$。

习题 8-12 Keep the Customer Satisfied, ACM/ICPC SWERC 2005,UVa1153

【分析】

首先对输入的所有订单按照截止日期从小到大排序。然后依次考虑每一个订单，同时记录当前已经安排好的所有订单以及做完这些订单完工时间，记为D。

对于一个新的订单i，如果D + q_i ≤ d_i，则将这个订单排进去，并且更新D += q_i。否则，看这个订单是不是比当前已经安排好的订单中用时最长的那个用时更短，如果是的话，就把这个任务替换进去，也要更新due。使用一个priority_queue来记录所有排进去的订单，方便快速找到并删除用时最长的订单。时间复杂度为$O(n \cdot logn)$。

下面考虑算法的正确性证明。初始结果一定是合法且最优的。类似数学归纳法，假设每一步处理完之后的结果一定合法且最优。每添加一个新订单，若能直接添加，结果一定合法且最优；如果不是最优，可以用前面某个丢掉的订单替换它而且还合法，这就说明上一步结果不是最优；添加时若超过时限，就删掉一个，删掉之前最费时的订单仍然合法，而且给后续订单留下的时间是最多的，依然最优。这样我们就可以证明最终结果合法且最优。

习题8-21 跳来跳去(Jumping Around, ACM/ICPC NEERC 2012, UVa1621)

你的任务是数轴上的0点出发，访问0, 1, 2, ..., n各一次，在任意点终止。需要用票才能从一个点到达另一个点。有三种票，跳跃长度为1, 2, 3，分别有a, b, c张（3<=a,b,c<=5000），且n=a+b+c。每张票只能用一次。输入保证有解。 比如a=3, b=4, c=3，则n=10，一种可能解为0->3->1->2->5->4->6->9->7->8->10，其中第1种票的3张分别用在1->2, 5->4, 7->8；第2种票的4张分别用在3->1, 4->6, 9->7, 8->10；第3种票的3张分别用在0->3, 2->5, 6->9。

【分析】

首先考虑c = 0的简单情况。那么从1开始先跳a-1次，把a用的就剩1次。此时剩下b+1个位置，且当前在其中最左边：

- b是偶数，则先向右跳b/2次，每一步长度为2，然后再向右跳一次到最右边，然后再向左跳b/2次刚好把b用完，并且所有位置都走过。比如a=3,b=4时，跳跃顺序就是0 1 2 4 6 7 5 3。
- b是奇数，则先向右跳b/2+1次(步长为2)，跳到最右边，然后向右跳1步，再向左跳b/2次把b用完，并且所有位置都走过。比如a=3,b=3时，跳跃顺序就是0 1 2 4 6 5 3。

下面考虑c ≠ 0的情况，考虑如何把c用完。 如果c%3 == 0, 则先向右跳c/3次(步长3)，然后向右跳一次(步长1)；向左跳c/3次(步长3)，向右跳1次(步长1)；向右跳c/3次(步长为3)。举例来说C=6，则跳跃顺序如下:

- 0 -> 3 -> 6 -> 7
- 7 -> 4 -> 1 -> 2
- 2 -> 5 -> 8

这样就把c用完，且跳过的位置形成一个连续区间。剩下就回到c=0的简单情况。 c%3 = 1和2的情况，请参考代码描述。