闷声刷大题解题报告

测试点1:

- 暴搜即可

测试点2：

- 由于所有的A都相同，所以可以先选A[1]，A[2]，……，A[k]，然后再选最小的k个B[i]
- sort一下即可

测试点3，4：

- 考虑建立费用流模型
- 把第i天拆成a_i和b_i，分别表示想和写，然后源向a_i连流量为1，费用为A[i]的边，a_i向b_j（满足i<=j）连流量为1，费用为0的边，b_i向汇连流量为1，费用为B[i]的边，因为只要做k道题，加个点限制下源的流量跑最小费用流即可

测试点5，6：

- 其实没有必要将a_i连向每个b_j，只需要a_i向b_i连流量为1，费用为0，b_i向b_{i+1}连流量为inf，费用为0的边，总边数为4n

满分做法：

- 如果把想题看成“（”，把写题看成“）”，则费用流每次增广其实就是加入一对“（）”或“）（”，且要保证当前这个括号序列合法,即序列的前缀和s_i在任意位置都大等于0

满分做法：

- 设左括号的位置是posa，右括号的位置是posb
- 假如选择加入“（）”，那么[posa,posb)的s_i要+1，假如选择加入“）（”，那么[posb,posa)的s_i要-1，且这一段减完后要>=0,即原来的[posb,posa)的最小值要>0
- 然后我们就要用线段树维护几个东西
- 前方高能，非战斗人员迅速撤退

满分做法：

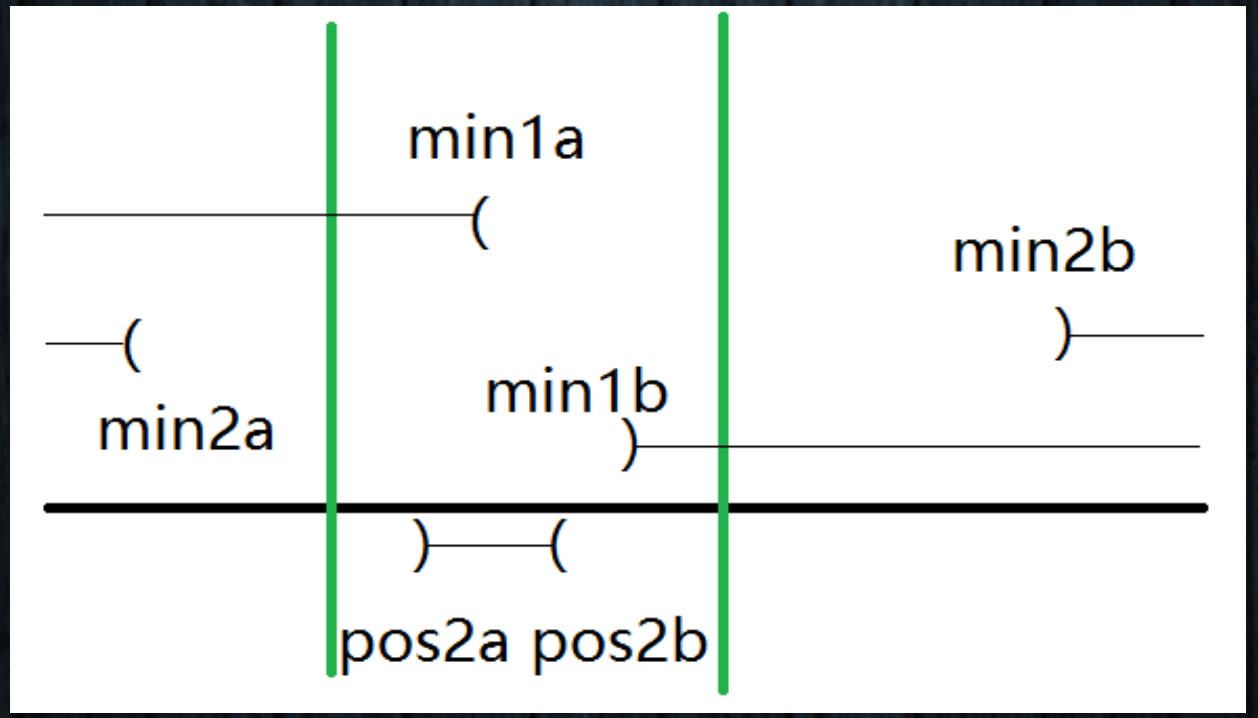
- case1a，case1b表示“（）”的最优解，“（”在case1a，“）”在case1b
- case2a，case2b表示“）（”的最优解，“（”在case2a，“）”在case2b，且[case2a,case2b)s_i的最小值大于当前区间si的最小值
- case3a，case3b表示“）（”的最优解，“（”在case2a，“）”在case2b

满分做法：

- min1a表示“（”的最小值的位置
- min1b表示“）”的最小值的位置
- min2a表示“（”的最小值的位置，且[st，min2a)si的最小值大于当前区间si的最小值（st表示当前区间的左端点）
- min2b表示“）”的最小值的位置，且[min2b，ed]si的最小值大于当前区间si的最小值（ed表示当前区间的右端点）
- minv表示当前区间si的最小值
- tag表示区间增量标记

满分做法：

- 假设绿线是minv所在的地方（显然可以不止一处），显然min1a到左端点可以经过绿线（也可以不经过），而min2a不可以，min1b、min2b同理，pos2b到pos2a之间不可以经过绿线，对于pos1a,pos1b,pos3a,pos3b都没有限制，如下图所示：



满分做法：

- 先考虑那些没有限制的标记的转移
- case1a，case1b的最小值可以用左儿子的case1a，case1b（即左儿子选一对“（）”），右儿子的case1a，case1b更新（即右儿子选一对“（）”），也可以用左儿子的min1a和右儿子的min1b更新（即用左儿子的“（”和右儿子的“）”来构成一对）
- case3同理
- min1a用左儿子和右儿子的min1a更新，min1b同理

满分做法：

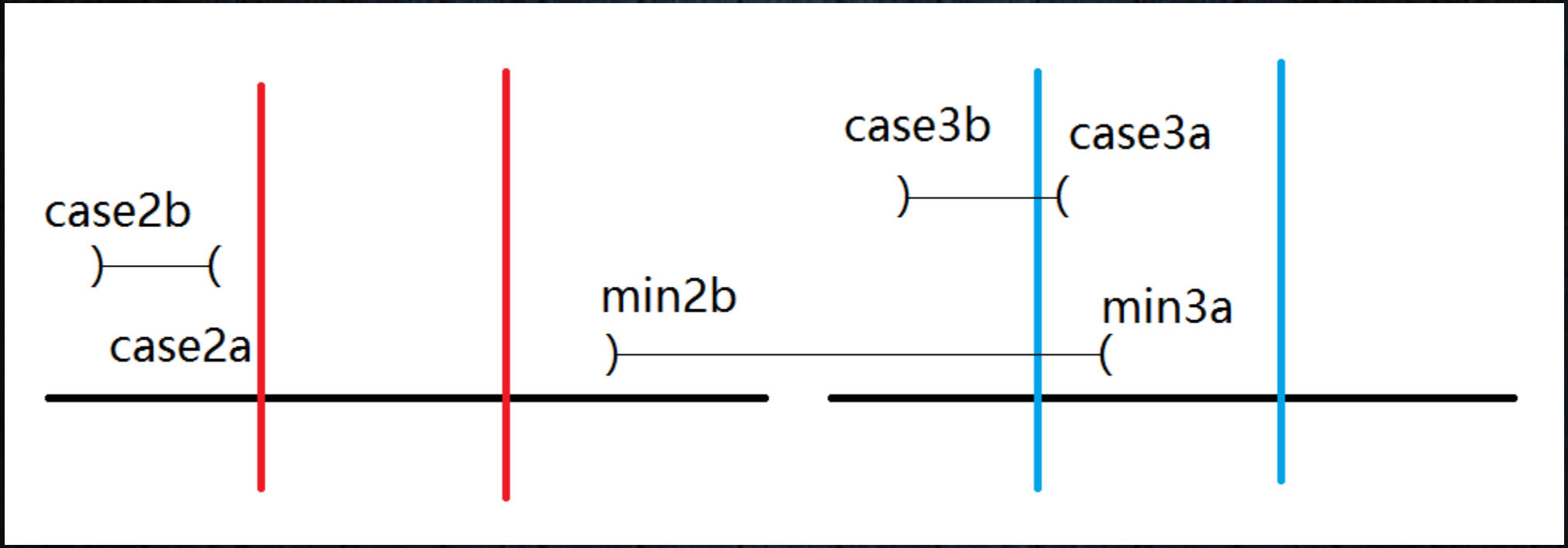
- 接下来就是有限制的标记的更新，我们要根据当前区间的最小值出现在哪里来分情况讨论：
- 下面的图表示左儿子的minv出现在红线所在的位置，右儿子的minv出现在蓝线所在的位置

满分做法：

- 情况一：
- 当左儿子的minv小于右儿子的minv，即当前区间的最小值只出现在红线所在的位置

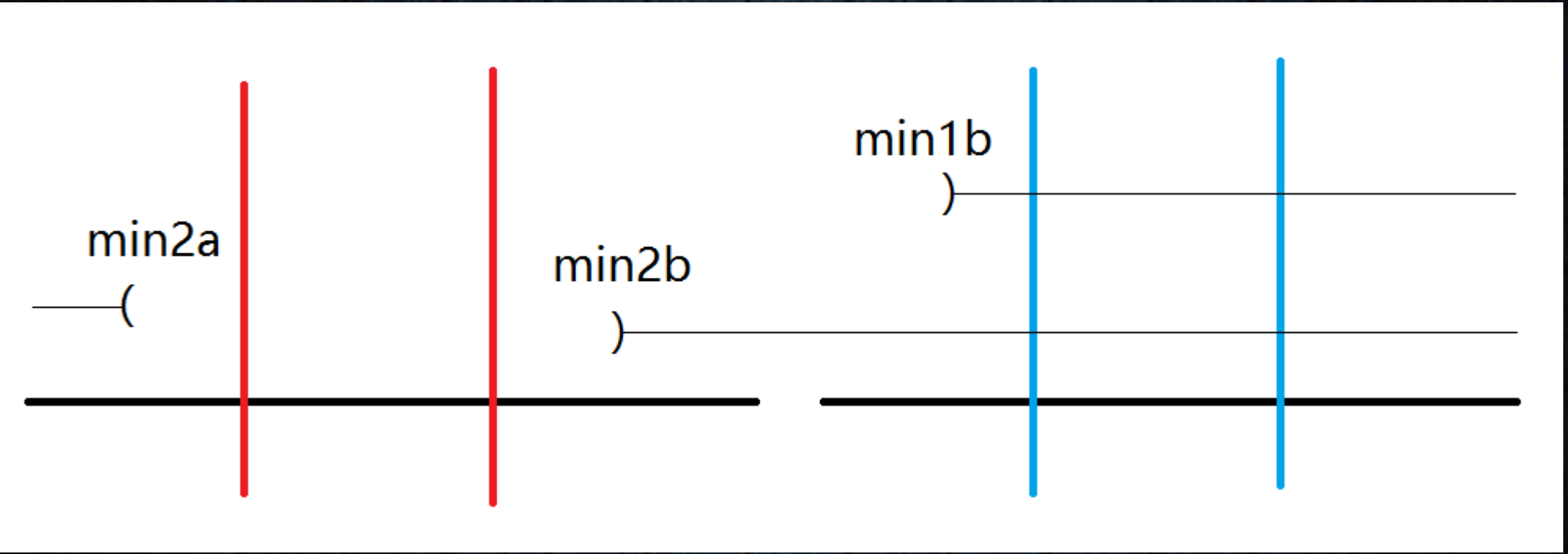
满分做法：

- 然后case2a和case2b可以由左儿子的case2a，case2b，右儿子的case3a，case3b更新，也可以用左儿子的min2b和右儿子的min1a更新，如下图所示：



满分做法：

- min2a只能用用左儿子的min2a更新
- min2b能用右儿子的min1b和左儿子的min2b更新
- 如图所示：



满分做法：

- 情况二：
- 当左儿子的minv大于右儿子的minv，即当前区间的最小值只出现在蓝线所在的位置
- 情况三：
- 当左儿子的minv等于于右儿子的minv，即当前区间的最小值只出现在红线和蓝线所在的位置
- 这两种情况类似的讨论（其实是懒得再画几张图了）

```
struct Data{int posa,posb;}tmp;
Data operator+(const Data &a,const Data &b){return A[a.posa]+B[a.posb]<A[b.posa]+B[b.posb]?a:b;}
struct Node{
    Data case1,case2,case3;
    int min1a,min1b,min2a,min2b,minv;
};
Node operator+(const Node &a,const Node &b){
    Node c;
    c.case1=a.case1+b.case1+(Data){a.min1a,b.min1b};
    c.case3=a.case3+b.case3+(Data){b.min1a,a.min1b};
    c.min1a=A[a.min1a]<A[b.min1a]?a.min1a:b.min1a;
    c.min1b=B[a.min1b]<B[b.min1b]?a.min1b:b.min1b;
    if (a.minv<b.minv){
        c.case2=a.case2+b.case3+(Data){b.min1a,a.min2b};
        c.min2a=a.min2a;
        c.min2b=B[a.min2b]<B[b.min1b]?a.min2b:b.min1b;
    }
    else if (a.minv>b.minv){
        c.case2=a.case3+b.case2+(Data){b.min2a,a.min1b};
        c.min2a=A[a.min1a]<A[b.min2a]?a.min1a:b.min2a;
        c.min2b=b.min2b;
    }
    else{
        c.case2=a.case2+b.case2+(Data){b.min2a,a.min2b};
        c.min2a=a.min2a;
        c.min2b=b.min2b;
    }
    c.minv=min(a.minv,b.minv);
    return c;
}
```

Think you for listening!