绍兴一中2016年暑期集训热身赛

Yuchong Pan

University of British Columbia

July 4, 2016

主要内容

Outline

题目大意

题目大意

- 在一个二维平面上，给定N个点，求在同一条直线上最多能有多少个点。

标准算法

- 点和斜率确定一条直线。

- 点和斜率确定一条直线。
- 枚举每一个点，计算其他点与这个点构成直线的斜率。

- 点和斜率确定一条直线。
- 枚举每一个点，计算其他点与这个点构成直线的斜率。
- 使用STL map或hash table等方法统计出同一种斜率对应的点的个数，取最大值。

- 点和斜率确定一条直线。
- 枚举每一个点，计算其他点与这个点构成直线的斜率。
- 使用STL map或hash table等方法统计出同一种斜率对应的点的个数，取最大值。
- 时间复杂度$O(N^2 logN)$（使用STL map），空间复杂度$O(N)$。

标准算法

- 为了避免double类型的精度误差，可以将斜率以分数的形式储存。

- 为了避免double类型的精度误差，可以将斜率以分数的形式储存。
- 可以用一个特殊的数来表示斜率不存在的直线，如INT_MAX。

- 为了避免double类型的精度误差，可以将斜率以分数的形式储存。
- 可以用一个特殊的数来表示斜率不存在的直线，如INT_MAX。
- 注意数据中可能出现重合点，单独统计即可。

Outline

题目大意

题目大意

- N只跳蚤在长度为$N+1$的横轴上。

题目大意

- N只跳蚤在长度为$N+1$的横轴上。
- 第i只跳蚤在坐标i上，重量为w_i。

题目大意

- N只跳蚤在长度为$N+1$的横轴上。
- 第i只跳蚤在坐标i上，重量为w_i。
- 跳蚤可以选择向左走或向右走。

- N只跳蚤在长度为$N+1$的横轴上。
- 第i只跳蚤在坐标i上，重量为w_i。
- 跳蚤可以选择向左走或向右走。
- 当两只跳蚤相遇时，大跳蚤吃掉小跳蚤，大跳蚤的重量增加小跳蚤的重量。

题目大意

- N只跳蚤在长度为$N+1$的横轴上。
- 第i只跳蚤在坐标i上，重量为w_i。
- 跳蚤可以选择向左走或向右走。
- 当两只跳蚤相遇时，大跳蚤吃掉小跳蚤，大跳蚤的重量增加小跳蚤的重量。
- 如果重量相同，左边的跳蚤吃掉右边的跳蚤。

题目大意

- N只跳蚤在长度为$N+1$的横轴上。
- 第i只跳蚤在坐标i上，重量为w_i。
- 跳蚤可以选择向左走或向右走。
- 当两只跳蚤相遇时，大跳蚤吃掉小跳蚤，大跳蚤的重量增加小跳蚤的重量。
- 如果重量相同，左边的跳蚤吃掉右边的跳蚤。
- 给出N和K，问第K只跳蚤最后留下的情况数。

子问题1

子问题1

- $w_i = i$，即w_i递增。

- $w_i = i$，即w_i递增。
- 跳蚤K只可能先向左，碰到端点后向右。

子问题1

- $w_i = i$，即w_i递增。
- 跳蚤K只可能先向左，碰到端点后向右。
- 考虑使用动态规划。

- $w_i = i$，即w_i递增。
- 跳蚤K只可能先向左，碰到端点后向右。
- 考虑使用动态规划。
- 从第K只跳蚤开始考虑，即假设数轴上只有K只跳蚤，求最后一只存货的情况数。

- $w_i = i$，即w_i递增。
- 跳蚤K只可能先向左，碰到端点后向右。
- 考虑使用动态规划。
- 从第K只跳蚤开始考虑，即假设数轴上只有K只跳蚤，求最后一只存货的情况数。
- 假设j是所有跳蚤中向左走的最右边一只，那么1到j所有跳蚤都会变成一个整体，重量为$\frac{j*(j+1)}{2}$。

- $w_i = i$，即w_i递增。
- 跳蚤K只可能先向左，碰到端点后向右。
- 考虑使用动态规划。
- 从第K只跳蚤开始考虑，即假设数轴上只有K只跳蚤，求最后一只存货的情况数。
- 假设j是所有跳蚤中向左走的最右边一只，那么1到j所有跳蚤都会变成一个整体，重量为$\frac{j*(j+1)}{2}$。
- 从K往前找到j，使得$\sum\limits_{i=j}^{K} w_i$大于$\frac{1}{2}\sum\limits_{i=1}^{K} w_i$。

- $w_i = i$，即w_i递增。
- 跳蚤K只可能先向左，碰到端点后向右。
- 考虑使用动态规划。
- 从第K只跳蚤开始考虑，即假设数轴上只有K只跳蚤，求最后一只存货的情况数。
- 假设j是所有跳蚤中向左走的最右边一只，那么1到j所有跳蚤都会变成一个整体，重量为$\frac{j*(j+1)}{2}$。
- 从K往前找到j，使得$\sum_{i=j}^{K} w_i$大于$\frac{1}{2}\sum_{i=1}^{K} w_i$。
- 此时j到$K-1$所有跳蚤的方向都是固定的，即向右。

- $w_i = i$，即w_i递增。
- 跳蚤K只可能先向左，碰到端点后向右。
- 考虑使用动态规划。
- 从第K只跳蚤开始考虑，即假设数轴上只有K只跳蚤，求最后一只存货的情况数。
- 假设j是所有跳蚤中向左走的最右边一只，那么1到j所有跳蚤都会变成一个整体，重量为$\frac{j*(j+1)}{2}$。
- 从K往前找到j，使得$\sum\limits_{i=j}^{K} w_i$大于$\frac{1}{2}\sum\limits_{i=1}^{K} w_i$。
- 此时j到$K-1$所有跳蚤的方向都是固定的，即向右。
- 跳蚤K的方向可以向左或向右。

- $w_i = i$，即w_i递增。
- 跳蚤K只可能先向左，碰到端点后向右。
- 考虑使用动态规划。
- 从第K只跳蚤开始考虑，即假设数轴上只有K只跳蚤，求最后一只存货的情况数。
- 假设j是所有跳蚤中向左走的最右边一只，那么1到j所有跳蚤都会变成一个整体，重量为$\frac{j*(j+1)}{2}$。
- 从K往前找到j，使得$\sum_{i=j}^{K} w_i$大于$\frac{1}{2}\sum_{i=1}^{K} w_i$。
- 此时j到$K-1$所有跳蚤的方向都是固定的，即向右。
- 跳蚤K的方向可以向左或向右。
- 此时$DP[K] = 2^{j+1}$。

子问题1

- 考虑$i > K$的情况。

子问题1

- 考虑$i > K$的情况。
- 设数组$a[j]$表示$\sum_{i=a[j]}^{j} w[i]$大于$\sum_{i=1}^{j-1} w[i]$的最大坐标。

- 考虑$i > K$的情况。
- 设数组$a[j]$表示$\sum_{i=a[j]}^{j} w[i]$大于$\sum_{i=1}^{j-1} w[i]$的最大坐标。
- 此时，$n-1$，$n-2$，$\dots$，$a[i]$均往右走，而n既可以往左走又可以往右走。

子问题1

- 考虑$i > K$的情况。
- 设数组$a[j]$表示$\sum_{i=a[j]}^{j} w[i]$大于$\sum_{i=1}^{j-1} w[i]$的最大坐标。
- 此时，$n-1$，$n-2$，...，$a[i]$均往右走，而n既可以往左走又可以往右走。
- 因此，$DP[n] = \sum_{i=a[i]}^{n-1} DP[i]$。

- 考虑$i > K$的情况。
- 设数组$a[j]$表示$\sum_{i=a[j]}^{j} w[i]$大于$\sum_{i=1}^{j-1} w[i]$的最大坐标。
- 此时，$n-1$，$n-2$，...，$a[i]$均往右走，而n既可以往左走又可以往右走。
- 因此，$DP[n] = \sum_{i=a[i]}^{n-1} DP[i]$。
- 至此，时间复杂度$O(N^2)$，空间复杂度$O(N)$。

子问题1

子问题1

- 用前缀和来优化DP转移。

- 用前缀和来优化DP转移。
- 即用 $DP\_sum[j]$ 表示 $\sum_{i=1}^{j} DP[i]$。

- 用前缀和来优化DP转移。
- 即用$DP_sum[j]$表示$\sum_{i=1}^{j} DP[i]$。
- 那么，$DP[i] = DP\_sum[n-1] - DP\_sum[a[i]-1]$。

- 用前缀和来优化DP转移。
- 即用$DP_sum[j]$表示$\sum_{i=1}^{j} DP[i]$。
- 那么，$DP[i] = DP\_sum[n-1] - DP\_sum[a[i]-1]$。
- 时间复杂度$O(N)$，空间复杂度$O(N)$。

子问题2

子问题2

- w_i无特殊限制。

子问题2

- w_i无特殊限制。
- 考虑使用区间动态规划进行预处理。

子问题2

- w_i无特殊限制。
- 考虑使用区间动态规划进行预处理。
- 令$F[i,j]$表示区间$[i,j]$作为一个整体向左走的方案数，$G[i,j]$表示区间$[i,j]$ 作为一个整体向右走的方案数。

- w_i无特殊限制。
- 考虑使用区间动态规划进行预处理。
- 令$F[i,j]$表示区间$[i,j]$作为一个整体向左走的方案数，$G[i,j]$表示区间$[i,j]$ 作为一个整体向右走的方案数。
- 转移分情况讨论即可。

- w_i无特殊限制。
- 考虑使用区间动态规划进行预处理。
- 令$F[i,j]$表示区间$[i,j]$作为一个整体向左走的方案数，$G[i,j]$表示区间$[i,j]$ 作为一个整体向右走的方案数。
- 转移分情况讨论即可。
- 转移时如果在最左端或最右端，则增加一种方案，即碰到端点后掉头。

子问题2

- 由于w_i无特殊限制，跳蚤K可以先向左走也可以先向右走。

子问题2

- 由于w_i无特殊限制，跳蚤K可以先向左走也可以先向右走。
- 分两种情况进行动态规划，分类讨论即可。

子问题2

- 由于w_i无特殊限制，跳蚤K可以先向左走也可以先向右走。
- 分两种情况进行动态规划，分类讨论即可。
- 同样需要注意转移时在两端的情况。

- 由于w_i无特殊限制，跳蚤K可以先向左走也可以先向右走。
- 分两种情况进行动态规划，分类讨论即可。
- 同样需要注意转移时在两端的情况。
- 具体细节详见代码。

- 由于w_i无特殊限制，跳蚤K可以先向左走也可以先向右走。
- 分两种情况进行动态规划，分类讨论即可。
- 同样需要注意转移时在两端的情况。
- 具体细节详见代码。
- 时间复杂度$O(N^2) - O(N^3)$，空间复杂度$O(N^2)$。

Outline

题目大意

题目大意

- 给定一个n个点m条边的无自环无重边的无向图。

题目大意

- 给定一个n个点m条边的无自环无重边的无向图。
- 可以删一条边，要求删边后的图是二分图。

题目大意

- 给定一个n个点m条边的无自环无重边的无向图。
- 可以删一条边，要求删边后的图是二分图。
- 问可以删哪些边。

题目大意

- 给定一个n个点m条边的无自环无重边的无向图。
- 可以删一条边，要求删边后的图是二分图。
- 问可以删哪些边。
- $n, m \leq 2 * 10^5$。

标准算法

标准算法

- 一个图是二分图的充要条件是这个图没有奇环。

- 一个图是二分图的充要条件是这个图没有奇环。
- 考虑*DFS*这个图，得到一棵*DFS*树。

标准算法

- 一个图是二分图的充要条件是这个图没有奇环。
- 考虑*DFS*这个图，得到一棵*DFS*树。
- 因为是无向图，所以非树边只会是返祖边。

- 一个图是二分图的充要条件是这个图没有奇环。
- 考虑*DFS*这个图，得到一棵*DFS*树。
- 因为是无向图，所以非树边只会是返祖边。
- 如果图不存在奇环，那么删任意一条边都是可行的。

标准算法

标准算法

- 考虑有奇环的情况。

- 考虑有奇环的情况。
- 考虑由一条返祖边构成的环。

- 考虑有奇环的情况。
- 考虑由一条返祖边构成的环。
- 如果这个环是奇环，那么必须从这个环中删掉一条边，否则整个图必然不是二分图。

- 考虑有奇环的情况。
- 考虑由一条返祖边构成的环。
- 如果这个环是奇环，那么必须从这个环中删掉一条边，否则整个图必然不是二分图。
- 如果只存在一个奇环，那么只能删掉环上的任意一条边。

标准算法

标准算法

- 考虑存在2个及以上奇环的情况。

- 考虑存在2个及以上奇环的情况。
- 如果存在2个及以上奇环，那么删任意一个奇环的返祖边显然无济于事，因为删返祖边只会破坏一个奇环。

- 考虑存在2个及以上奇环的情况。
- 如果存在2个及以上奇环，那么删任意一个奇环的返祖边显然无济于事，因为删返祖边只会破坏一个奇环。
- 因此必须删树边。

标准算法

- 观察出如下结论:

- 观察出如下结论:
- 有且仅有2条属于奇环的返祖边的环必然是偶环。

- 观察出如下结论:
- 有且仅有2条属于奇环的返祖边的环必然是偶环。
- 有且仅有2条属于偶环的返祖边的环必然是偶环。

- 观察出如下结论:
- 有且仅有2条属于奇环的返祖边的环必然是偶环。
- 有且仅有2条属于偶环的返祖边的环必然是偶环。
- 有且仅有1条属于奇环的返祖边和1条属于偶环的返祖边的环必然是奇环。

标准算法

- 下面证明以下条件是可行解的充要条件:

- 下面证明以下条件是可行解的充要条件:
- 这条边必须同时属于所有的奇环。

- 下面证明以下条件是可行解的充要条件:
- 这条边必须同时属于所有的奇环。
- 这条边不属于任何偶环。

标准算法

- 先证明必要性。

- 先证明必要性。
- 如果删掉一条边后，存在一个奇环没有被破坏，那么显然不合法。

- 先证明必要性。
- 如果删掉一条边后，存在一个奇环没有被破坏，那么显然不合法。
- 所以删掉的边必须同时属于所有的奇环。

- 先证明必要性。
- 如果删掉一条边后，存在一个奇环没有被破坏，那么显然不合法。
- 所以删掉的边必须同时属于所有的奇环。
- 如果一条边属于一个偶环，那么必然意味着，有一条属于偶环的返祖边跨越了它。

- 先证明必要性。
- 如果删掉一条边后，存在一个奇环没有被破坏，那么显然不合法。
- 所以删掉的边必须同时属于所有的奇环。
- 如果一条边属于一个偶环，那么必然意味着，有一条属于偶环的返祖边跨越了它。
- 同时，它属于所有奇环，所以也必然存在一条属于奇环的返祖边跨越了它。

- 先证明必要性。
- 如果删掉一条边后，存在一个奇环没有被破坏，那么显然不合法。
- 所以删掉的边必须同时属于所有的奇环。
- 如果一条边属于一个偶环，那么必然意味着，有一条属于偶环的返祖边跨越了它。
- 同时，它属于所有奇环，所以也必然存在一条属于奇环的返祖边跨越了它。
- 由结论3，拥有1条属于奇环的返祖边和1条属于偶环的返祖边的环必然是奇环。

- 先证明必要性。
- 如果删掉一条边后，存在一个奇环没有被破坏，那么显然不合法。
- 所以删掉的边必须同时属于所有的奇环。
- 如果一条边属于一个偶环，那么必然意味着，有一条属于偶环的返祖边跨越了它。
- 同时，它属于所有奇环，所以也必然存在一条属于奇环的返祖边跨越了它。
- 由结论3，拥有1条属于奇环的返祖边和1条属于偶环的返祖边的环必然是奇环。
- 与假设矛盾，因此这条边不能属于任何一个偶环。

标准算法

- 同样可以证明，这个约束是可行解的充分条件。

- 同样可以证明，这个约束是可行解的充分条件。
- 于是得到了最终的结论:

- 同样可以证明，这个约束是可行解的充分条件。
- 于是得到了最终的结论:
- 一条树边合法，当且仅当它属于所有奇环，不属于任何偶环。

标准算法

标准算法

- 考虑维护每条树边属于奇环和偶环的个数。

- 考虑维护每条树边属于奇环和偶环的个数。
- 在DFS时记录深度，找到一条返祖边，即可判断这条返祖边所在环的奇偶性。

- 考虑维护每条树边属于奇环和偶环的个数。
- 在*DFS*时记录深度，找到一条返祖边，即可判断这条返祖边所在环的奇偶性。
- 直观来看，可以用*LCT*或树链剖分维护这个环上的树边。

- 考虑维护每条树边属于奇环和偶环的个数。
- 在*DFS*时记录深度，找到一条返祖边，即可判断这条返祖边所在环的奇偶性。
- 直观来看，可以用*LCT*或树链剖分维护这个环上的树边。
- 但是考虑到问题是离线的，可以在树上进行差分维护。

- 考虑维护每条树边属于奇环和偶环的个数。
- 在DFS时记录深度，找到一条返祖边，即可判断这条返祖边所在环的奇偶性。
- 直观来看，可以用LCT或树链剖分维护这个环上的树边。
- 但是考虑到问题是离线的，可以在树上进行差分维护。
- 时间复杂度$O(N+M)$。

生成数据

生成数据

- 对于绝大多数随机数据，问题都是无解的。

生成数据

- 对于绝大多数随机数据，问题都是无解的。
- 考虑构造数据。

生成数据

- 对于绝大多数随机数据，问题都是无解的。
- 考虑构造数据。
- 我们需要满足有一定数量的边属于所有奇环而不属于任一偶环。

生成数据

- 对于绝大多数随机数据，问题都是无解的。
- 考虑构造数据。
- 我们需要满足有一定数量的边属于所有奇环而不属于任一偶环。
- 考虑构造一条长链和一些环，使得一些奇环横跨长链的绝大部分，一些偶环横跨长链的小部分。

Outline

参考资料

参考资料

- LeetCode 149: Max Points on a Line.

参考资料

- LeetCode 149: Max Points on a Line.
- ACM-ICPC EC-Final 2015 F: Hungry Game of Ants.

参考资料

- LeetCode 149: Max Points on a Line.
- ACM-ICPC EC-Final 2015 F: Hungry Game of Ants.
- Codeforces Beta Round 19 E: Fairy.

- LeetCode 149: Max Points on a Line.
- ACM-ICPC EC-Final 2015 F: Hungry Game of Ants.
- Codeforces Beta Round 19 E: Fairy.
- Codeforces题目泛做解题报告, 许昊然.