Hello World ! --sgu

记忆化搜索 多维搜索 启发题

题意：m个气球n个人吹。第i个人一分钟能吹Ai个气球然后至少休息Bi分钟才能吹下一分钟。问最少需要多少分钟吹完所有气球？(Bi <= 4)

思路：f[sum][a1][a2][a3][a4]表示吹sum个气球，前1,2,3,4分钟吹的人分别是a1,a2,a3,a4需要的最少时间。当题目中出现一维非常小时，此处往往就是问题的突破口。

完全就是按雷神的代码来写的.

启发：虽然知道Bi<=4是突破口，但自己想的时候也没有什么好的搜索方法。。想过f[i][j]表示第i个气球让志愿者j吹，但是搜起来的话要处理的细节搞不清楚，就写不下去。

说说这道题的记忆化搜解法的启发吧。

平常真的对多维的东西不太敢想，以至于就算AC了也觉得，下次再遇到这种题，还是会跪。因为没能把题目和该算法联系起来。好像知道“其所以然”...

int main()

{

scanf("%d%d", &M, &n);

for(int i = 1; i <= n; i++) {

scanf("%d%d", &work[i], &rest[i]);

}

memset(f, -1, sizeof(f));

printf("%d\n", dfs(M, 0, 0, 0, 0));

return 0;

}

int dfs(int sum, int a, int b, int c, int d)

{

if(sum <= 0) return 0;

int &result = f[sum][a][b][c][d];

if(result != -1) return result;

result = INF;

result = min(result, 1 + dfs(sum, 0, a, b, c));//

for(int i = 1; i <= n; i++) if(work[i]) {

if(i == a && rest[i] > 0) continue;

if(i == b && rest[i] > 1) continue;

if(i == c && rest[i] > 2) continue;

if(i == d && rest[i] > 3) continue;

result = min(result, 1 + dfs(sum - work[i], i, a, b, c));

}

return result;

}

最长路关键点 可行点 单调队列 好题

题意：平面上若干点（xi,yi),点i能到达点j当且仅当xi < xj, yi < yj

那这样的话我们会有很多条最长路，现在要输输出所有可行点和关键点。

可行点定义：某条最长路经过该点

关键点定义：所有最长路都经过该点

思路：可行点好求，方法有几种。问题是关键点。

neko提出关键点其实就是DAG图上的割点，太麻烦，想别的(之后证明这个想法是错的)

仔细看neko根据样例画的点图，一起想到了”拦截导弹II”，感觉很类似，于是就根据

这个往下想。

按x升序，y降序排好序 （好巧妙）

从左往右来一次严格最长上升子序列dp1，从右往左来一次严格最长下降子序列dp2

若dp1[i] + dp2[i] == lis + 1说明i点是可行点

若dp1[i]这个值只出现过一次，那i点是关键点

想复用LCS过程，看VJ上有个人的方法挺不错，求完第一次LCS后，逆序再取反，再来

一次LCS就出来dp2了，不过现在的dp2也是逆序的.

虚拟出个无穷大点做汇点~

好吧，我二分又写挫了，还是neko帮调的

扩展：如果场景改为一个有向图呢？可行点和关键点定义相同。

现在反倒关键点好求了，就是拓扑分层一边，发现某层只有一个点的话就是关键点

可行点的话~~随便YY吧

下面贴个neko刚整理的二分xxx code:

template<class Cmp>

int LIS(Cmp cmp) {

static int m, end[N];

m = 0;

forl(i, 0, n) {

int pos = lower\_bound(end, end+m, a[i], cmp)-end;

end[pos] = a[i], m += pos==m;

}

return m;

}

int main() {

cout << LIS(less<int>()) << endl;

cout << LIS(less\_equal<int>()) << endl;

cout << LIS(greater<int>()) << endl;

cout << LIS(greater\_equal<int>()) << endl;