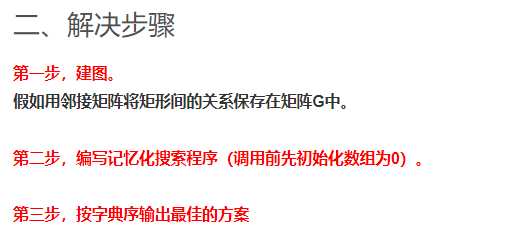
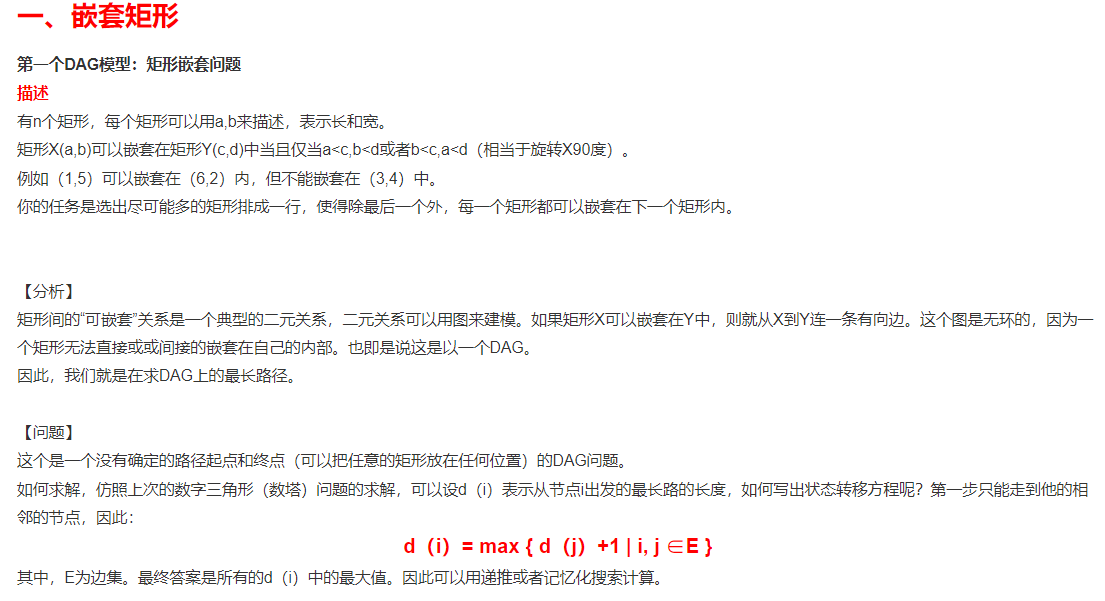
DAG图上解决问题有几种思路，一种是用拓扑排序的方式进行递推，一种则是下面的这种记忆化搜索



//记忆化搜索来完成的动态转移

int dp(int i)//这一段记忆化搜索是重点

{

int j;

if(d[i] > 0)

return d[i]; // 如果已经计算过，直接返回其值

d[i] = 1; // 否则，置一，递推计算

for(j = 1; j <= n; j++)

if(G[i][j]) // 如果图存在，即是满足可嵌套

if(d[i] <=dp(j)+1) // 如果存在可嵌套的节点d(j)加一后其值大于d(i)

d[i]=dp(j)+1; // 则使d[i]更新

return d[i]; // 返回d[i]

}

void print\_ans(int i)

{

int j;

printf("%d ", i); // 第一次i代表最长路的起点节点，以后均代表从该节点开始的路径

for(j = 1; j <= n; j++)

if(G[i][j] && d[i] == d[j]+1) // 如果该图满足可嵌套，且d[i] = d[j] +1

{

print\_ans(j); // 立即输出从节点j开始的路径

break;

}

}

int main()

{

int i, j, t, ans, best;

scanf("%d", &n); // n表示矩形的数目

// 初始化矩形长宽参数，并初次调整长宽顺序

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

{

scanf("%d%d", &x[i], &y[i]); // 依次输入矩形的边长信息

if(x[i] > y[i])

{

t = x[i]; x[i] = y[i]; y[i] = t; // 保证X[]存的是长，Y[]存的是宽

}

}

memset(G, 0, sizeof(G)); // 数组清零

for(i = 1; i <= n; i++) // 建图

for(j = 1; j <= n; j++)

if(x[i] < x[j] && y[i] < y[j]) G[i][j] = 1; // 如果第i个矩形的长宽均小于第j个，使图相应的值为1

ans = 0;

for(i = 1; i <= n; i++) // 依次递推所有的的节点

if(dp(i) > ans)

{

best = i; // best 是最小字典序

ans = dp(i);

}

printf("ans=%d\n", ans); // 表示最长路长度

print\_ans(best);

printf("\n");

while(1);

return 0 ;

}