2016-7-1

关于**优先队列**(其实没什么可写的233)

priority\_queue< int, vector<int>, less<int> > q;

less 可用 greater 替代

如果要自定义则需要

struct cmp{

bool operater()(myType a, myType b){

retrun a < b;

}

}

priority\_queue< int, vector<int>, cmp > q;

2016-7-2

可以用map来记录点状态，适用于无限大地图但只有有限个点可以使用的情况（没办法开状态数组）

map<pair<int,int>,int>m;

map中用自定义类型时，需要重载 <

struct node{

int x, y, z;

node(int a, int b, int c):x(a),y(b),z(c){}

bool operator < (const node & node1) const{

if (x == node1.x && y == node1.y)

return z < node1.z;

else if (x == node1.x)

return y < node1.y;

return x < node1.x;

}

};

2016-7-11

复习了一下图论的简单知识

最短路Dijkstra算法使用优先队列实现

SPFA算法，用来寻找负权环，使用队列保存每次更新的点，若某个点更新次数超过n次，则出现负权

2016-7-12

学习了字典树，具体在第一场个人赛中有代码。

2016-7-13

Longlong在做位移运算时需要声明类型 : numLL << k;

Next\_permutation() 可以快速的求出全排列

2016-7-15

未优化的next数组可以求解循环节，对于一个字符串， m - next[m] 是他的最小循环节，同时, m-next[next[m]] 等也是循环节

**二分图：**

补充定义和定理：

最大匹配数：最大匹配的匹配边的数目

最小点覆盖数：选取最少的点，使任意一条边至少有一个端点被选择

最大独立数：选取最多的点，使任意所选两点均不相连

最小路径覆盖数：对于一个 DAG（有向无环图），选取最少条路径，使得每个顶点属于且仅属于一条路径。路径长可以为 0（即单个点）。

定理1：最大匹配数 = 最小点覆盖数（这是 Konig 定理）

定理2：最大匹配数 = 最大独立数

定理3：最小路径覆盖数 = 顶点数 - 最大匹配数

2016-7-16

匈牙利算法建边的时候建单向边即可，即左集合到右集合，linker数组保存了匹配边的端点。

当不知道每个点在哪测的集合时可以建双向边，然后总结果除以二。

用塔尖算法求割点割边，low数组记录当前结点所有子节点通过非父子边（就是不能沿着父亲到儿子的边，只能从别的边）所能到达的最浅的深度，dep记录节点的深度。

对于节点u，若u是根节点，则记录其子数的数量，若大于1，则必然是割点。（在dfs中每通过根节点延伸一次，则说明多了一棵字数）

若u不是根节点，且v是u的子结点，若low[v] >= dep[u]，即u的子节点不能通过除u以外的边到达u的父节点，则u是一个割点。若low[v] > dep[u] ，则u到v是一条割边。

因为a^n-1 = 1 (mod b) gcd(a, b) = 1;

a^(n-2) 就是a模b的乘法逆元

2016-7-18

2016-7-20

线段的离散化 将一堆线段离散 例：

对于线段： （1 ，2） （ 5，10） （1，12） 可以离散化成 （1 2） （3 4） （1 5）

就是按照值的大小映射到1-k中，记录每一个线段中的点的id值，id值相同的点属于同一条线段。

2016-8-3

卡特兰数：

Cn = (2n)! / ((n + 1)! \* n!)

所有的奇卡特兰数都满足n = 2^k – 1，所有其他的卡特兰数都是偶数

卡特兰数的应用：

对于一个由ｎ个Ｘ和ｎ个Ｙ构成的字符串中，满足任意前缀中的Ｘ的个数大于等于Ｙ的个数的序列的数量。

具体的参见数学中的维基百科对卡特兰数的总结

2016-8-9

匈牙利算法求最大匹配的时候双向建边，然后从每一个点出发寻找增广路，所得结果是最大匹配数的二倍