|  |
| --- |
| Meow |
| Meow模板 |
| Meow |

|  |
| --- |
| Ruosen Lee  2016-10-17 |

目录

[图论 2](#_Toc464473923)

[2-SAT 2](#_Toc464473924)

[最短路 4](#_Toc464473925)

[A-Star 4](#_Toc464473926)

[Bellman-Ford 5](#_Toc464473927)

[Bellman-Ford(检查负环) 5](#_Toc464473928)

[Dijkstra(次短路) 6](#_Toc464473929)

[Dijkstra(记录路径) 6](#_Toc464473930)

[Dijkstra-pq 7](#_Toc464473931)

[Floyd 7](#_Toc464473932)

[Spfa 8](#_Toc464473933)

[Spfa-slf-pq 8](#_Toc464473934)

[生成树 9](#_Toc464473935)

[Prim-pq 9](#_Toc464473936)

[Kruskal 9](#_Toc464473937)

[曼哈顿最小生成树 10](#_Toc464473938)

[Prim次小生成树 12](#_Toc464473939)

[有向图最小生成树 13](#_Toc464473940)

[最小生成树计数 14](#_Toc464473941)

[二分图 16](#_Toc464473942)

[匈牙利 16](#_Toc464473943)

[HK 16](#_Toc464473944)

[KM 17](#_Toc464473945)

[网络流 18](#_Toc464473946)

[Dinic 18](#_Toc464473947)

[Ford-Flukerson 20](#_Toc464473948)

[费用流 20](#_Toc464473949)

[欧拉路径 22](#_Toc464473950)

[Fleury 22](#_Toc464473951)

[全局最小割 23](#_Toc464473952)

[差分约束 24](#_Toc464473953)

[数据结构 26](#_Toc464473954)

[LCA 26](#_Toc464473955)

[RMQ 27](#_Toc464473956)

[差分前缀和 27](#_Toc464473957)

[线段树 28](#_Toc464473958)

[树状数组 29](#_Toc464473959)

[可持久化数据结构 29](#_Toc464473960)

[静态主席树 29](#_Toc464473961)

[动态主席树 30](#_Toc464473962)

[可持久化线段树 32](#_Toc464473963)

[划分树 33](#_Toc464473964)

[莫队算法 35](#_Toc464473965)

[莫队分块 35](#_Toc464473966)

[树上莫队 36](#_Toc464473967)

[树链剖分 36](#_Toc464473968)

[动态规划 38](#_Toc464473969)

[树DP 38](#_Toc464473970)

[经典 38](#_Toc464473971)

[删点 39](#_Toc464473972)

[树上背包 39](#_Toc464473973)

[应用：树上任意点能到达的最远距离 40](#_Toc464473974)

[区间DP 42](#_Toc464473975)

[数位DP 42](#_Toc464473976)

[可能的树分治模板 43](#_Toc464473977)

[字符串 45](#_Toc464473978)

[AC自动机 45](#_Toc464473979)

[KMP 46](#_Toc464473980)

[数学 47](#_Toc464473981)

[Ploya定理 47](#_Toc464473982)

[SG博弈 47](#_Toc464473983)

[逆元处理求组合数(MOD是质数) 48](#_Toc464473984)

[组合数 49](#_Toc464473985)

[快速线性素数筛 49](#_Toc464473986)

[计算几何 50](#_Toc464473987)

[点与矩阵最小距离 50](#_Toc464473988)

[点在多边形内 50](#_Toc464473989)

[多边形与圆面积交 51](#_Toc464473990)

[矩形面积并 53](#_Toc464473991)

[凸多边形面积并 55](#_Toc464473992)

[旋转卡壳 57](#_Toc464473993)

[圆的面积并(辛普森积分法) 57](#_Toc464473994)

[杂七杂八 59](#_Toc464473995)

[LIS 59](#_Toc464473996)

[斯坦纳树 59](#_Toc464473997)

[归并排序求逆序对 61](#_Toc464473998)

[环境 62](#_Toc464473999)

[\_vimrc配置文件 62](#_Toc464474000)

[CB配置 62](#_Toc464474001)

# 图论

## 2-SAT

|  |
| --- |
| 在实际问题中，2-SAT问题在大多数时候表现成以下形式：有N对物品，每对物品中必须选取一个，也只能选取一个，并且它们之间存在某些限制关系（如某两个物品不能都选，某两个物品不能都不选，某两个物品必须且只能选一个，某个物品必选）等，这时，可以将每对物品当成一个布尔值（选取第一个物品相当于0，选取第二个相当于1），如果所有的限制关系最多只对两个物品进行限制，则它们都可以转化成9种基本限制关系，从而转化为2-SAT模型。  其实2-SAT问题的建模是和实际问题非常相似的。  建立一个2N阶的有向图，其中的点分为N对，每对点表示布尔序列A的一个元素的0、1取值（以下将代表A[i]的0取值的点称为i，代表A[i]的1取值的点称为i'）。显然每对点必须且只能选取一个。然后，图中的边具有特定含义。若图中存在边<i, j>，则表示若选了i必须选j。  【O(NM)算法：求字典序最小的解】  根据2-SAT建成的图中边的定义可以发现，若图中i到j有路径，则若i选，则j也要选；或者说，若j不选，则i也不能选；  因此得到一个很直观的算法：  （1）给每个点设置一个状态V，V=0表示未确定，V=1表示确定选取，V=2表示确定不选取。称一个点是已确定的当且仅当其V值非0。设立两个队列Q1和Q2，分别存放本次尝试选取的点的编号和尝试不选的点的编号。  （2）若图中所有的点均已确定，则找到一组解，结束，否则，将Q1、Q2清空，并任选一个未确定的点i，将i加入队列Q1，将i'加入队列Q2；  （3）找到i的所有后继。对于后继j，若j未确定，则将j加入队列Q1；若j'（这里的j'是指与j在同一对的另一个点）未确定，则将j'加入队列Q2；  （4）遍历Q2中的每个点，找到该点的所有前趋（这里需要先建一个补图），若该前趋未确定，则将其加入队列Q2；  （5）在（3）（4）步操作中，出现以下情况之一，则本次尝试失败，否则本次尝试成功：  <1>某个已被加入队列Q1的点被加入队列Q2；  <2>某个已被加入队列Q2的点被加入队列Q1;  <3>某个j的状态为2；  <4>某个i'或j'的状态为1或某个i'或j'的前趋的状态为1；  （6）若本次尝试成功，则将Q1中的所有点的状态改为1，将Q2中所有点的状态改为2，转（2），否则尝试点i'，若仍失败则问题无解。  该算法的时间复杂度为O(NM)（最坏情况下要尝试所有的点，每次尝试要遍历所有的边），但是在多数情况下，远远达不到这个上界。  具体实现时，可以用一个数组vst来表示队列Q1和Q2。设立两个标志变量i1和i2（要求对于不同的i，i1和i2均不同，这样可以避免每次尝试都要初始化一次，节省时间），若vst[i]=i1则表示i已被加入Q1，若vst[i]=i2则表示i已被加入Q2。不过Q1和Q2仍然是要设立的，因为遍历（BFS）的时候需要队列，为了防止重复遍历，加入Q1（或Q2）中的点的vst值必然不等于i1（或i2）。中间一旦发生矛盾，立即中止尝试，宣告失败。  该算法虽然在多数情况下时间复杂度到不了O(NM)，但是综合性能仍然不如下面的O(M)算法。不过，该算法有一个很重要的用处：求字典序最小的解！  如果原图中的同一对点编号都是连续的（01、23、45……）则可以依次尝试第0对、第1对……点，每对点中先尝试编号小的，若失败再尝试编号大的。这样一定能求出字典序最小的解（如果有解的话），因为一个点一旦被确定，则不可更改。  如果原图中的同一对点编号不连续（比如03、25、14……）则按照该对点中编号小的点的编号递增顺序将每对点排序，然后依次扫描排序后的每对点，先尝试其编号小的点，若成功则将这个点选上，否则尝试编号大的点，若成功则选上，否则（都失败）无解。 |
| 1. const int INF = 0x7F7F7F7F; 2. const int MAXN = 1000 + 10; 3. const int MAXM = 1000000 + 10; 4. struct Edge{ int to, next; }; 5. Edge es[MAXM]; 6. int head[MAXN], low[MAXN], dfn[MAXN], belong[MAXN], a[MAXN], b[MAXN]; 7. bool insta[MAXN]; 8. int n, m, cnt, index; 9. stack<int> sta; 10. void add( int u, int v ) { 11. es[cnt].to = v; es[cnt].next = head[u]; head[u] = cnt++; 12. return ; 13. } 14. void tarjan( int u ) { 15. int v; 16. dfn[u] = low[u] = index++; 17. sta.push( u ); 18. insta[u] = true; 19. for( int i = head[u]; ~i; i = es[i].next ) { 20. v = es[i].to; 21. if( dfn[v] == -1 ) { 22. tarjan( v ); 23. low[u] = min( low[u], low[v] ); 24. } else if( insta[v] ) { 25. low[u] = min( low[u], dfn[v] ); 26. } 27. } 28. if( dfn[u] == low[u] ) { 29. do { 30. v = sta.top(); sta.pop(); 31. insta[v] = false; 32. belong[v] = cnt; 33. } while( u != v ); 34. ++cnt; 35. } 36. return ; 37. } 38. int main() { 39. scanf( "%d%d", &n, &m ); 40. memset( head, -1, sizeof( head ) ); 41. memset( dfn, -1, sizeof( dfn ) ); 42. memset( low, -1, sizeof( low ) ); 43. memset( insta, false, sizeof( insta ) ); 44. memset( belong, -1, sizeof( belong ) ); 45. cnt = 0; 46. for( int i = 0; i < m; ++i ) { 47. scanf( "%d%d", a + i, b + i ); 48. if( a[i] > b[i] ) swap( a[i], b[i] ); 49. } 50. for( int i = 0; i < m; ++i ) { 51. for( int j = i + 1; j < m; ++j ) { 52. if( ( a[i] < a[j] && b[i] < b[j] && a[j] < b[i] ) || ( a[i] > a[j] && b[i] > b[j] && b[j] > a[i] ) ) { 53. add( i \* 2, j \* 2 + 1 ); 54. add( j \* 2, i \* 2 + 1 ); 55. add( i \* 2 + 1, j \* 2 ); 56. add( j \* 2 + 1, i \* 2 ); 57. } 58. } 59. } 60. index = 1; cnt = 1; 61. for( int i = 0; i < m \* 2; ++i ) { 62. if( dfn[i] == -1 ) tarjan( i ); 63. } 64. bool flag = true; 65. for( int i = 0; i < m \* 2; ++i ) { 66. if( belong[i] == belong[i ^ 1] ) { 67. flag = false; 68. break; 69. } 70. } 71. if( flag ) printf( "panda is telling the truth...\n" ); 72. else printf( "the evil panda is lying again\n" ); 73. return 0; 74. } |

## 最短路

### A-Star

|  |
| --- |
| /\*  h为当前代价  spfa为逆向搜索，寻找每个节点的估值函数g的值  \*/   1. const int INF = 0x7F7F7F7F; 2. const int MAXN = 10000 + 10; 3. const int MAXE = 1000000 + 10; 4. struct Edge { int to, cost, next; }; 5. Edge es[MAXE]; 6. struct Node { int u, f, g; }; 7. int head[MAXN], h[MAXN]; 8. bool vis[MAXN]; 9. int n, m, k, cnt; 10. vector<int> path; 11. struct cmp { 12. bool operator() ( const Node &a, const Node &b ) { 13. return a.f > b.f; 14. } 15. }; 16. void add( int u, int v, int w ) { 17. es[cnt].to = v; es[cnt].cost = w; es[cnt].next = head[u]; head[u] = cnt++; 18. return ; 19. } 20. void spfa() { 21. queue<int> que; 22. memset( vis, false, sizeof( vis ) ); 23. memset( h, 0x7F, sizeof( h ) ); 24. que.push( n ); h[n] = 0; vis[n] = true; 25. while( !que.empty() ) { 26. int u = que.front(); que.pop(); 27. for( int i = head[u]; ~i; i = es[i].next ) { 28. int v = es[i].to; 29. if( h[v] > h[u] + es[i].cost ) { 30. h[v] = h[u] + es[i].cost; 31. if( !vis[v] ) { 32. vis[v] = true; 33. que.push( v ); 34. } 35. } 36. } 37. vis[u] = false; 38. } 39. return ; 40. } 41. void astar() { 42. priority\_queue<Node, vector<Node>, cmp> que; 43. Node tmp = { 1, h[1], 0 }; 44. que.push( tmp ); 45. for( int cur = 0; cur < k && !que.empty(); ) { 46. tmp = que.top(), que.pop(); 47. int u = tmp.u, g = tmp.g; 48. if( u == n ) { ++cur; path.push\_back( g ); } 49. for( int j = head[u]; ~j; j = es[j].next ) { 50. tmp.u = es[j].to; 51. tmp.g = g + es[j].cost; 52. tmp.f = tmp.g + h[tmp.u]; 53. que.push( tmp ); 54. } 55. } 56. return ; 57. } 58. int main() { 59. int a, b, c; 60. k = 2; 61. while( ~scanf( "%d%d", &n, &m ) ) { 62. memset( head, -1, sizeof( head ) ); 63. cnt = 0; 64. for( int i = 1; i <= m; ++i ) { 65. scanf( "%d%d%d", &a, &b, &c ); 66. add( a, b, c ); 67. add( b, a, c ); 68. } 69. spfa(); 70. astar(); 71. printf( "%d\n", path[k - 1] ); 72. } 73. return 0; 74. } |

### Bellman-Ford

|  |
| --- |
| 1. #define INF 0x7FFFFFFF 2. #define MAX\_V 1000 3. #define MAX\_E 1000000 + 10 4. struct edge { int from, to, cost; }; 5. edge es[MAX\_E]; 6. int d[MAX\_V]; 7. int V, E; 8. void bf( int s ) { 9. for( int i = 0; i < V; ++i ) d[i] = INF; 10. d[s] = 0; 11. while( true ) { 12. bool flag = false; 13. for( int i = 0; i < E; ++i ) { 14. edge e = es[i]; 15. if( d[e.from] != INF && d[e.to] > d[e.from] + e.cost ) { 16. d[e.to] = d[e.from] + e.cost; 17. flag = true; 18. } 19. } 20. if( !flag ) break; 21. } 22. return ; 23. } |

### Bellman-Ford(检查负环)

|  |
| --- |
| 1. #define INF 0x7FFFFFFF 2. #define MAX\_V 1000 3. #define MAX\_E 1000000 + 10 4. struct edge { int from, to, cost; }; 5. edge es[MAX\_E]; 6. int d[MAX\_V]; 7. int V, E; 8. bool bf\_nagative( int s ) { 9. memset( d, 0, sizeof( d ) ); 10. for( int i = 0; i < V; ++i ) { 11. for( int j = 0; j < E; ++j ) { 12. edge e = es[j]; 13. if( d[e.to] > d[e.from] + e.cost ) { 14. d[e.to] = d[e.from] + e.cost; 15. if( i == V - 1 ) return true; 16. } 17. } 18. } 19. return false; 20. } |

### Dijkstra(次短路)

|  |
| --- |
| 1. #define N 110 2. #define PII pair<int, int> 3. #define INF 0x7FFFFFFF 4. struct edge { 5. int to, cost; 6. edge( int t, int c ) { to = t; cost = c; } 7. }; 8. vector<edge> G[N]; 9. int V; 10. int dis[N], dis2[N]; 11. void dijikstra( int s ) { 12. priority\_queue<PII, vector<PII>, greater<PII> > pq; 13. for( int i = 0; i < V; ++i ) dis[i] = dis2[i] = INF; 14. dis[s] = 0; 15. pq.push( PII( dis[s], s ) ); 16. while( !pq.empty() ) { 17. PII p = pq.top(); 18. pq.pop(); 19. int v = p.second, d = p.first; 20. if( dis2[v] < d ) continue; 21. for( int i = 0; i < G[v].size(); ++i ) { 22. edge &e = G[v][i]; 23. int d2 = d + e.cost; 24. if( d2 < dis[e.to] ) { 25. swap( d2, dis[e.to] ); 26. pq.push( PII( dis[e.to], e.to ) ); 27. } 28. if( d2 < dis2[e.to] && d2 > dis[e.to] ) { 29. dis[e.to] = d2; 30. pq.push( PII( dis2[e.to], e.to ) ); 31. } 32. } 33. } 34. return ; 35. } 36. int main() { 37. int m, n; 38. while( ~scanf( "%d%d" ,&m, &n ) && ( m || n ) ) { 39. for( int i = 0; i < n; ++i ) G[i].clear(); 40. V = n; 41. int u, v, w; 42. for( int i = 0; i < m; ++i ) { 43. scanf( "%d%d%d", &u, &v, &w ); 44. G[u - 1].push\_back( edge( v - 1, w ) ); 45. G[v - 1].push\_back( edge( u - 1, w ) ); 46. } 47. dijikstra( 0 ); 48. cout << dis[n - 1] << endl; 49. } 50. return 0; 51. } |

### Dijkstra(记录路径)

|  |
| --- |
| 1. #define N 1010 2. #define INF 0x7FFFFFFF 3. int d[N], vis[N], pre[N]; 4. int cost[N][N]; 5. int V; 6. void dijkstra( int s ) { 7. for( int i = 1; i <= V; ++i ) { 8. d[i] = INF; 9. vis[i] = 0; 10. pre[i] = -1; 11. } 12. d[s] = 0; 13. while( true ) { 14. int v = -1; 15. for( int u = 1; u <= V; ++u ) { 16. if( !vis[u] && ( v == -1 || d[u] < d[v] ) ) v = u; 17. } 18. if( v == -1 ) break; 19. vis[v] = 1; 20. for( int u = 1; u <= V; ++u ) { 21. if( d[u] > d[v] + cost[v][u] ) { 22. d[u] = d[v] + cost[v][u]; 23. pre[u] = v; 24. } 25. } 26. } 27. return ; 28. } |

### Dijkstra-pq

|  |
| --- |
| 1. #define INF 0x7FFFFFFF 2. #define MAX\_V 1000 3. #define MAX\_E 1000000 + 10 4. typedef pair<int, int> PII; 5. struct edge { int to, cost; }; 6. vector<edge> G[MAX\_V]; 7. int d[MAX\_V]; 8. int V; 9. void dijskra( int s ) { 10. priority\_queue<PII, vector<PII>, greater<PII> > pq; 11. for( int i = 0; i < V; ++i ) d[i] = INF; 12. d[s] = 0; 13. pq.push( PII( 0, s ) ); 14. while( !pq.empty() ) { 15. PII p = pq.top(); pq.pop(); 16. int v = p.second; 17. if( d[v] < p.first ) continue; 18. for( int i = 0; i < G[v].size(); ++i ) { 19. edge e = G[v][i]; 20. if( d[e.to] > d[v] + e.cost ) { 21. d[e.to] = d[v] + e.cost; 22. pq.push( PII( d[e.to], e.to ) ); 23. } 24. } 25. } 26. return ; 27. } |

### Floyd

|  |
| --- |
| 1. const int INF = 0x7F7F7F7F; 2. const int MAXN = 1e3 + 10; 3. int d[MAXN][MAXN]; 4. int n; 5. void floyd() { 6. for( int k = 0; k < n; ++k ) { 7. for( int i = 0; i < n; ++i ) 8. for( int j = 0; j < n; ++j ) 9. d[i][j] = min( d[i][j], d[i][k] + d[k][j] ); 10. } 11. return ; 12. } |

### Spfa

|  |
| --- |
| 1. const int INF = 0x7F7F7F7F; 2. const int MAXN = 1e3 + 10; 3. int cost[MAXN][MAXN]; 4. int d[MAXN]; 5. bool used[MAXN]; 6. int n; 7. void spfa( int s ) { 8. queue<int> q; 9. for( int i = 0; i < n; ++i ) { d[i] = INF; used[i] = false; } 10. d[s] = 0; used[s] = true; 11. q.push( s ); 12. while( !q.empty() ) { 13. int u = q.front(); q.pop(); 14. used[u] = false; 15. for( int i = 0; i < n; ++i ) { 16. if( d[u] + cost[u][i] < d[i] ) { 17. d[i] = d[u] + cost[u][i]; 18. if( !used[i] ) { 19. used[i] = true; 20. q.push( i ); 21. } 22. } 23. } 24. } 25. return ; 26. } |

### Spfa-slf-pq

|  |
| --- |
| 1. const int INF = 0x7F7F7F7F; 2. const int MAXN = 1e3 + 10; 3. int cost[MAXN][MAXN]; 4. int d[MAXN]; 5. bool used[MAXN]; 6. int num[MAXN]; 7. int n; 8. struct cmp { 9. bool operator() ( int x, int y ) { 10. return d[x] > d[y]; 11. } 12. }; 13. bool spfa\_slf\_pq( int s ) { 14. priority\_queue<int, vector<int>, cmp > pq; 15. for( int i = 0; i < n; ++i ) { d[i] = INF; used[i] = false; num[i] = 0; } 16. d[s] = 0; used[s] = true; ++num[s]; 17. pq.push( s ); 18. while( !pq.empty() ) { 19. int u = pq.top(); pq.pop(); 20. used[s] = false; 21. for( int i = 0; i < n; ++i ) { 22. if( d[u] + cost[u][i] < d[i] ) { 23. d[i] = d[u] + cost[u][i]; 24. if( !used[i] ) { 25. ++num[i]; 26. if( num[i] > n ) return false; 27. pq.push( i ); 28. used[i] = true; 29. } 30. } 31. } 32. } 33. return true; 34. } |

## 生成树

### Prim-pq

|  |
| --- |
| 1. #define INF 0x7FFFFFFF 2. #define MAX\_V 1000 3. #define MAX\_E 1e6 4. struct edge { to, cost }; 5. typedef pair<int, int> PII; 6. vector<edge> G[MAX\_V]; 7. int mincost[MAX\_V]; 8. int V; 9. int prim() { 10. int res = 0; 11. priority\_queue<PII, vector<PII>, greater<PII> > pq; 12. for( int i = 0; i < V; ++i ) mincost[i] = INF; 13. mincost[0] = 0; pq.push( P( 0, 0 ) ); 14. while( !pq.empty() ) { 15. PII tmp = pq.top(); pq.pop(); 16. int v = tmp.second; res += v; 17. if( mincost[v] < tmp.first ) continue; 18. for( int i = 0; i < G[v].size(); ++i ) { 19. edge e = G[v][i]; 20. if( mincost[e.to] > e.cost ) { 21. mincost[e.to] = e.cost; 22. pq.push( mincost[e.to], e.to ); 23. } 24. } 25. } 26. return res; 27. } |

### Kruskal

|  |
| --- |
| 关于次小生成树  但有一种更简单的方法：先求最小生成树T，枚举添加不在T中的边，则添加后一定会形成环。找到环上边值第二大的边(即环中属于T中的最大边)，把它删掉，计算当前生成树的权值，取所有枚举修改的生成树的最小值，即为次小生成树。  这种方法在实现时有更简单的方法：首先求最小生成树T，然后从每个结点u遍历最小生成树T，用一个二维数组max[u][v]记录结点u到结点v的路劲上边的最大值(即最大边的值)。然后枚举不在T中的边(u,v)，计算T- max[u][v] + w(u,v)的最小值，即为次小生成树的权值。显然，这种方法的时间复杂度为O(n^2 + e)。 |
| 1. const int INF = 0x7F7F7F7F; 2. const int MAXN = 1e3 + 10; 3. const int MAXM = 1e6 + 10; 4. struct edge { int u, v, cost; }; 5. edge es[MAX\_V]; 6. int V, E; 7. int father[MAX\_V], mrank[MAX\_V]; 8. int mfind( int x ) { 9. if( x != father[x] ) 10. father[x] = mfind( father[x] ); 11. return father[x]; 12. } 13. void munion( int x, int y ) { 14. if( mrank[x] > mrank[y] ) father[y] = x; 15. else { 16. if( mrank[x] == mrank[y] ) ++mrank[y]; 17. father[x] = y; 18. } 19. return ; 20. } 21. bool cmp( const edge& e1, const edge& e2 ) { 22. return e1.cost > e2.cost; 23. } 24. int kruskal() { 25. int res = 0; 26. sort( es, es + E, cmp ); 27. for( int i = 0; i < V; ++i ) { father[i] = i; mrank[i] = 0; } 28. for( int i = 0; i < E; ++i ) { 29. int x = mfind( es[i].u ); 30. int y = mfind( es[i].v ); 31. if( x != y ) { munion( x, y ); res += es[i].cost; } 32. } 33. return res; 34. } |

### 曼哈顿最小生成树

|  |
| --- |
| 1. typedef long long LL; 2. const int INF = 0x3F3F3F3F; 3. const int MAXN = 1000000 + 10; 4. struct Point { 5. int x, y, id; 6. bool operator < ( const Point &p ) const { 7. return x == p.x ? y < p.y : x < p.x; 8. } 9. } poi[MAXN]; 10. struct BIT { 11. int minVal, pos; 12. void init() { minVal = INF; pos = -1; } 13. } bit[MAXN << 2]; 14. struct Edge { 15. int u, v, cost; 16. bool operator < ( const Edge &e ) const { 17. return cost < e.cost; 18. } 19. } es[MAXN << 2]; 20. int fa[MAXN], a[MAXN], b[MAXN]; 21. int n, k, cnt; 22. int mfind( int x ) { return x == fa[x] ? x : fa[x] = mfind( fa[x] ); } 23. void add( int u, int v, int w ) { 24. es[cnt].u = u; es[cnt].v = v; es[cnt].cost = w; ++cnt; 25. return ; 26. } 27. int lowbit( int x ) { return x & -x; } 28. void update( int i, int val, int pos ) { 29. while( i ) { 30. if( val < bit[i].minVal ) { 31. bit[i].minVal = val; 32. bit[i].pos = pos; 33. } 34. i -= lowbit( i ); 35. } 36. return ; 37. } 38. int ask( int i, int m ) { 39. int minVal = INF, pos = -1; 40. while( i <= m ) { 41. if( bit[i].minVal < minVal ) { 42. minVal = bit[i].minVal; 43. pos = bit[i].pos; 44. } 45. i += lowbit( i ); 46. } 47. return pos; 48. } 49. int dist( const Point &a, const Point &b ) { 50. return abs( a.x - b.x ) + abs( a.y - b.y ); 51. } 52. int MHT( int k ) { 53. cnt = 0; 54. for( int dir = 0; dir < 4; ++dir ) { 55. if( dir == 1 || dir == 3 ) { 56. for( int i = 0; i < n; ++i ) swap( poi[i].x, poi[i].y ); 57. } 58. if( dir == 2 ) { 59. for( int i = 0; i < n; ++i ) poi[i].x \*= -1; 60. } 61. sort( poi, poi + n ); 62. for( int i = 0; i < n; ++i ) a[i] = b[i] = poi[i].y - poi[i].x; 63. sort( b, b + n ); 64. int ncnt = unique( b, b + n ) - b; 65. for( int i = 1; i <= ncnt; ++i ) bit[i].init(); 66. for( int i = n - 1; i >= 0; --i ) { 67. int pos = lower\_bound( b, b + ncnt, a[i] ) - b + 1; 68. int ans = ask( pos, ncnt ); 69. if( ans != -1 ) add( poi[i].id, poi[ans].id, dist( poi[i], poi[ans] ) ); 70. update( pos, poi[i].x + poi[i].y, i ); 71. } 72. } 73. sort( es, es + cnt ); 74. for( int i = 0; i < n; ++i ) fa[i] = i; 75. for( int i = 0; i < cnt; ++i ) { 76. int u = es[i].u, v = es[i].v; 77. int x = mfind( u ), y = mfind( v ); 78. if( x != y ) { 79. --k; 80. fa[x] = y; 81. if( k == 0 ) return es[i].cost; 82. } 83. } 84. return 0; 85. } 86. int main() { 87. while( ~scanf( "%d%d", &n, &k ) ) { 88. for( int i = 0; i < n; ++i ) { 89. scanf( "%d%d", &poi[i].x, &poi[i].y ); 90. poi[i].id = i; 91. } 92. printf( "%d\n", MHT( n - k ) ); 93. } 94. return 0; 95. } |

### Prim次小生成树

|  |
| --- |
| 1. typedef pair<int, int> PII; 2. const int INF = 0x7F7F7F7F; 3. const int MAXN = 1000 + 10; 4. PII poi[MAXN]; 5. double dis[MAXN][MAXN], path[MAXN][MAXN], mincost[MAXN]; 6. int ren[MAXN], pre[MAXN]; 7. bool vis[MAXN], used[MAXN][MAXN]; 8. int n; 9. double dist( const int i, const int j ) { 10. double dx = poi[i].first - poi[j].first; 11. double dy = poi[i].second - poi[j].second; 12. return sqrt( dx \* dx + dy \* dy ); 13. } 14. double prim() { 15. double ret = 0; 16. memset( used, false, sizeof( used ) ); 17. memset( vis, false, sizeof( vis ) ); 18. memset( path, 0, sizeof( path ) ); 19. for( int i = 0; i < n; ++i ) { mincost[i] = INF; pre[i] = 0; } 20. mincost[0] = 0; 21. while( true ) { 22. int v = -1; 23. for( int u = 0; u < n; ++u ) if( !vis[u] && ( v == -1 || mincost[u] < mincost[v] ) ) v = u; 24. if( v == -1 ) break; 25. used[pre[v]][v] = used[v][pre[v]] = true; 26. ret += mincost[v]; 27. vis[v] = true; 28. for( int u = 0; u < n; ++u ) { 29. if( vis[u] && v != u ) path[u][v] = path[v][u] = max( path[u][pre[v]], mincost[v] ); 30. if( !vis[u] && mincost[u] > dis[u][v] ) { 31. mincost[u] = dis[u][v]; 32. pre[u] = v; 33. } 34. } 35. } 36. return ret; 37. } 38. int main() { 39. int t; 40. scanf( "%d", &t ); 41. while( t-- ) { 42. scanf( "%d", &n ); 43. for( int i = 0; i < n; ++i ) { 44. scanf( "%d%d%d", &poi[i].first, &poi[i].second, ren + i ); 45. for( int j = 0; j < i; ++j ) dis[i][j] = dis[j][i] = dist( i, j ); 46. dis[i][i] = 0; 47. } 48. double tmp = prim(); 49. double ans = -1; 50. for( int i = 0; i < n; ++i ) { 51. for( int j = 0; j < n; ++j ) if( i != j ) { 52. if( used[i][j] ) ans = max( ans, ( ren[i] + ren[j] ) / ( tmp - dis[i][j] ) ); 53. else ans = max( ans, ( ren[i] + ren[j] ) / ( tmp - path[i][j] ) ); 54. } 55. } 56. printf( "%.2f\n", ans ); 57. } 58. return 0; 59. } |

### 有向图最小生成树

|  |
| --- |
| 1. const double INF = 0x3F3F3F3F; 2. const int MAXN = 100 + 10; 3. const int MAXE = 100000 + 10; 4. struct edge{ int u, v; double cost; }; 5. edge es[MAXE]; 6. int ID[MAXN], vis[MAXN], pre[MAXN], x[MAXN], y[MAXN]; 7. double IN[MAXN]; 8. int n, m, cnt; 9. void add( int u, int v, double c ) { 10. es[cnt].u = u; es[cnt].v = v; es[cnt].cost = c; ++cnt; 11. return ; 12. } 13. double direct\_MST( int root ) { 14. double ans = 0; 15. while( true ) { 16. memset( ID, -1, sizeof( ID ) ); 17. memset( vis, -1, sizeof( vis ) ); 18. for( int i = 0; i < MAXN; ++i ) IN[i] = INF; 19. for( int i = 0; i < m; ++i ) { 20. int u = es[i].u; 21. int v = es[i].v; 22. if( es[i].cost < IN[v] && u != v ) { 23. IN[v] = es[i].cost; 24. pre[v] = u; 25. } 26. } 27. for( int i = 0; i < n; ++i ) { 28. if( i == root ) continue; 29. if( IN[i] == INF ) return -1; 30. } 31. int tv = 0; 32. IN[root] = 0;// pre[root] = root; 33. for( int i = 0; i < n; ++i ) { 34. ans += IN[i]; 35. int v = i; 36. while( vis[v] != i && ID[v] == -1 && v != root ) { 37. vis[v] = i; 38. v = pre[v]; 39. } 40. if( v != root && ID[v] == -1 ) { 41. for( int u = pre[v]; u != v; u = pre[u] ) { 42. ID[u] = tv; 43. } 44. ID[v] = tv++; 45. } 46. } 47. if( !tv ) break; 48. for( int i = 0; i < n; ++i ) { 49. if( ID[i] == -1 ) ID[i] = tv++; 50. } 51. for( int i = 0; i < m; ++i ) { 52. int v = es[i].v; 53. es[i].u = ID[es[i].u]; 54. es[i].v = ID[es[i].v]; 55. if( es[i].u != es[i].v ) 56. es[i].cost -= IN[v]; 57. } 58. n = tv; 59. root = ID[root]; 60. } 61. return ans; 62. } 63. double dis( int i, int j ) { 64. double dx = abs( x[i] - x[j] ); 65. double dy = abs( y[i] - y[j] ); 66. return sqrt( dx \* dx + dy \* dy ); 67. } 68. int main() { 69. int a, b; 70. while( ~scanf( "%d%d", &n, &m ) ) { 71. cnt = 0; 72. for( int i = 0; i < n; ++i ) scanf( "%d%d", x + i, y + i ); 73. for( int i = 0; i < m; ++i ) { 74. scanf( "%d%d", &a, &b ); 75. --a; --b; 76. add( a, b, dis( a, b ) ); 77. } 78. double ans = direct\_MST( 0 ); 79. if( ans < 0 ) puts( "poor snoopy" ); 80. else printf( "%.2f\n", ans ); 81. } 82. return 0; 83. } |

### 最小生成树计数

|  |
| --- |
| 1. typedef long long LL; 2. const int MAXN = 1000 + 10; 3. const int MAXE = 100000 + 10; 4. struct Edge { int u, v, w; }; 5. Edge es[MAXE]; 6. int fa[MAXN], ka[MAXN]; 7. LL g[MAXN][MAXN], c[MAXN][MAXN]; 8. bool vis[MAXN]; 9. int n, m, mod; 10. vector<int> vec[MAXN]; 11. int mfind( int x, int \*f ) { 12. return x == f[x] ? x : f[x] = mfind( f[x], f ); 13. } 14. LL det( LL a[][MAXN], int n ) { 15. for( int i = 0; i < n; ++i ) { 16. for( int j = 0; j < n; ++j ) 17. a[i][j] %= mod; 18. } 19. int ret = 1; 20. for( int i = 1; i < n; ++i ) { 21. for( int j = i + 1; j < n; ++j ) { 22. while( a[j][i] ) { 23. LL t = a[i][i] / a[j][i]; 24. for( int k = i; k < n; ++k ) 25. a[i][k] = ( a[i][k] - a[j][k] \* t ) % mod; 26. for( int k = i; k < n; ++k ) 27. swap( a[i][k], a[j][k] ); 28. ret = -ret; 29. } 30. } 31. if( a[i][i] == 0 ) return 0; 32. ret = ( ret \* a[i][i] ) % mod; 33. } 34. return ( ret + mod ) % mod; 35. } 36. bool cmp( const Edge &a, const Edge &b ) { 37. return a.w < b.w; 38. } 39. void gao() { 40. sort( es, es + m, cmp ); 41. for( int i = 1; i <= n; ++i ) { fa[i] = i; vis[i] = false; } 42. LL pre = -1, ans = 1; 43. for( int k = 0; k <= m; ++k ) { 44. if( es[k].w != pre || k == m ) { 45. for( int i = 1; i <= n; ++i ) { 46. if( vis[i] ) { 47. LL u = mfind( i, ka ); 48. vec[u].push\_back( i ); 49. vis[i] = false; 50. } 51. } 52. for( int i = 1; i <= n; ++i ) { 53. if( vec[i].size() > 1 ) { 54. memset( c, 0, sizeof( c ) ); 55. int len = vec[i].size(); 56. for( int j = 0; j < len; ++j ) { 57. for( int k = j + 1; k < len; ++k ) { 58. int a1 = vec[i][j], b1 = vec[i][k]; 59. c[j][k] = ( c[k][j] -= g[a1][b1] ); 60. c[j][j] += g[a1][b1]; c[k][k] += g[a1][b1]; 61. } 62. } 63. LL ret = det( c, len ); 64. ans = ( ans \* ret ) % mod; 65. for( int j = 0; j < len; ++j ) fa[vec[i][j]] = i; 66. } 67. } 68. for( int i = 1; i <= n; ++i ) { 69. ka[i] = fa[i] = mfind( i, fa ); 70. vec[i].clear(); 71. } 72. if( k == m ) break; 73. pre = es[k].w; 74. } 75. int u = es[k].u, v = es[k].v; 76. int a1 = mfind( u, fa ), b1 = mfind( v, fa ); 77. if( a1 == b1 ) continue; 78. vis[a1] = vis[b1] = true; 79. ka[mfind( a1, ka )] = mfind( b1, ka ); 80. ++g[a1][b1]; ++g[b1][a1]; 81. } 82. bool flag = false; 83. for( int i = 2; i <= n && !flag; ++i ) { 84. if( ka[i] != ka[i - 1] ) flag = true; 85. } 86. if( !m ) flag = true; 87. printf( "%I64d\n", flag ? 0 : ans % mod ); 88. return ; 89. } 90. int main() { 91. while( ~scanf( "%d%d%d", &n, &m, &mod ) && n + m + mod ) { 92. memset( g, 0, sizeof( g ) ); 93. for( int i = 1; i <= n; ++i ) vec[i].clear(); 94. for( int i = 0; i < m; ++i ) scanf( "%d%d%d", &es[i].u, &es[i].v, &es[i].w ); 95. gao(); 96. } 97. return 0; 98. } |

## 二分图

### 匈牙利

|  |
| --- |
| 1. #define INF 0x7FFFFFFF 2. #define MAX\_V 1000 3. #define MAX\_E 1000000 + 10 4. vector<int> G[MAX\_V]; 5. bool used[MAX\_V]; 6. int match[MAX\_V]; 7. int VX; 8. int findPath( int k ) { 9. for( int i = 0; i < G[k].size(); ++i ) { 10. int t = G[k][i]; 11. if( !used[t] ) { 12. used[t] = true; 13. if( match[i] == -1 || findPath( match[i] ) ) { 14. match[t] = k; 15. return 1; 16. } 17. } 18. } 19. return 0; 20. } 21. int hungary() { 22. int res; 23. memset( match, -1, sizeof( match ) ); 24. for( int i = 0; i < VX; ++i ) { 25. memset( used, false, sizeof( used ) ); 26. res += findPath( i ); 27. } 28. return res; 29. } |

### HK

|  |
| --- |
| 1. #define INF 0x7FFFFFFF 2. #define MAX\_V 1000 3. #define MAX\_E 1000000 + 10 4. vector<int> G[MAX\_V]; 5. int VX; 6. int dx[MAX\_V], dy[MAX\_V]; 7. int cx[MAX\_V], cy[MAX\_V]; 8. int mindis; 9. bool mask[MAX\_V]; 10. bool searchPath() { 11. queue<int> q; 12. memset( dx, -1, sizeof( dx ) ); 13. memset( dy, -1, sizeof( dy ) ); 14. mindis = INF; 15. for( int i = 0; i < VX; ++i ) { 16. if( cx[i] == -1 ) { 17. dx[i] = 0; 18. q.push( i ); 19. } 20. } 21. while( !q.empty() ) { 22. int t = q.front(); q.pop(); 23. if( dx[t] > mindis ) break; 24. for( int i = 0; i < G[t].size(); ++i ) { 25. int v = G[t][i]; 26. if( dy[v] == -1 ) { 27. dy[v] = dx[t] + 1; 28. if( cy[v] == -1 ) mindis = dy[v]; 29. else { 30. dx[cy[v]] = dy[v] + 1; 31. q.push( v ); 32. } 33. } 34. } 35. } 36. return mindis != INF; 37. } 38. int findPath( int u ) { 39. for( int i = 0; i < G[u].size(); ++i ) { 40. int v = G[u][i]; 41. if( !mask[v] && dy[v] == dx[u] + 1 ) { 42. mask[v] = true; 43. if( cy[v] != -1 && dy[v] == mindis ) continue; 44. else { 45. cx[u] = v; cy[v] = u; 46. return 1; 47. } 48. } 49. } 50. return 0; 51. } 52. int hk() { 53. int res = 0; 54. memset( cx, -1, sizeof( cx ) ); 55. memset( cy, -1, sizeof( cy ) ); 56. while( searchPath() ) { 57. memset( mask, 0, sizeof mask ); 58. for( int i = 0; i < VX; ++i ) { 59. if( cx[i] == -1 ) 60. res += findPath( i ); 61. } 62. } 63. return res; 64. } |

### KM

|  |
| --- |
| 1. const int INF = 0x7F7F7F7F; 2. int nmap[305][305]; 3. int lx[305],ly[305]; 4. bool x[305],y[305]; 5. int link[305]; 6. int n; // n可能要改成n, m 7. bool dfs( int u ) { 8. int i; 9. x[u] = true; 10. for( i = 1; i <= n; ++i ) { 11. if( lx[u] + ly[i] == nmap[u][i] && !y[i] ) { 12. y[i] = true; 13. if( link[i] == -1 || dfs( link[i] ) ) { 14. link[i] = u; 15. return true; 16. } 17. } 18. } 19. return false; 20. } 21. void KM() { 22. int i, j, k; 23. memset( x, 0, sizeof( x ) ); 24. memset( y, 0, sizeof( y ) ); 25. memset( link, -1, sizeof( link ) ); 26. for( i = 1; i <= n; ++i ) lx[i] = INF; 27. memset( ly, 0, sizeof( ly ) ); 28. for( k = 1; k <= n; ++k ) { 29. while( true ) { 30. memset( x, 0, sizeof( x ) ); 31. memset( y, 0, sizeof( y ) ); 32. if( dfs( k ) ) break; 33. int d = INF; 34. for( i = 1; i <= n; ++i ) { 35. if( x[i] ) 36. for( j = 1; j <= n; ++j ) 37. if( !y[j] && lx[i] + ly[j] - nmap[i][j] < d ) 38. d = lx[i] + ly[j] - nmap[i][j]; 39. } 40. for( i = 1; i <= n; ++i ) if( x[i] ) lx[i] = lx[i] - d; 41. for( i = 1; i <= n; ++i ) if( y[i] ) ly[i] = ly[i] + d; 42. } 43. } 44. return ; 45. } 46. int main() { 47. int i, j, k; 48. while( ~scanf( "%d", &n ) ) { 49. // nmap可能要初始化，求最小值时清成-INF即0x80808080 50. for( i = 1; i <= n; ++i ) 51. for( j = 1; j <= n; ++j ) 52. scanf( "%d", &nmap[i][j] ); 53. KM(); 54. int ans = 0; 55. for( i = 1; i <= n; ++i ) ans = ans + nmap[link[i]][i]; 56. printf( "%d\n", ans ); 57. } 58. return 0; 59. } |

## 网络流

### Dinic

|  |
| --- |
| add双向边连续建图，cnt从0开始，^操作确定相邻两边 |
| 1. typedef int MyType; 2. const int INF = 0x3F3F3F3F; 3. const int MAXN = 1000 + 10; 4. const int MAXE = 100000 + 10; 5. struct Edge { int to, next; MyType cap; }; 6. Edge es[MAXE]; 7. int head[MAXN], cur[MAXN], level[MAXN], que[MAXN]; 8. int n, F, D, cnt, src, des; 9. void add( int u, int v, MyType c ) { 10. es[cnt].to = v; es[cnt].cap = c; es[cnt].next = head[u]; head[u] = cnt++; 11. es[cnt].to = u; es[cnt].cap = 0; es[cnt].next = head[v]; head[v] = cnt++; 12. return ; 13. } 14. bool bfs() { 15. int mf, me; 16. memset( level, 0, sizeof( level ) ); 17. mf = me = 0; 18. que[me++] = src; 19. level[src] = 1; 20. while( mf < me ) { 21. int u = que[mf++]; 22. for( int i = head[u]; ~i; i = es[i].next ) { 23. int v = es[i].to; 24. if( level[v] == 0 && es[i].cap > 0 ) { 25. level[v] = level[u] + 1; 26. que[me++] = v; 27. } 28. } 29. } 30. return ( level[des] != 0 ); 31. } 32. MyType dfs( int u, MyType f ) { 33. if( u == des || f == 0 ) return f; 34. MyType flow = 0; 35. for( int &i = cur[u]; ~i; i = es[i].next ) { 36. Edge &e = es[i]; 37. if( e.cap > 0 && level[e.to] == level[u] + 1 ) { 38. MyType d = dfs( e.to, min( f, e.cap ) ); 39. if( d > 0 ) { 40. e.cap -= d; 41. es[i ^ 1].cap += d; 42. flow += d; 43. f -= d; 44. if( f == 0 ) break; 45. } else level[e.to] = -1; 46. } 47. } 48. return flow; 49. } 50. MyType dinic() { 51. MyType ret = 0; 52. while( bfs() ) { 53. for( int i = src; i <= des; ++i ) { 54. cur[i] = head[i]; 55. } 56. ret += dfs( src, INF ); 57. } 58. return ret; 59. } 60. int main() { 61. return 0; 62. } |

### Ford-Flukerson

|  |
| --- |
| 1. #define MAXV 10010 2. #define INF 0x7FFFFFFF 3. struct edge { int to, cap, rev; }; 4. vector<edge> G[MAXV]; 5. bool vis[MAXV]; 6. void add\_edge( int f, int t, int c ) { 7. G[f].push\_back( ( edge ){ t, c, G[t].size() } ); 8. G[t].push\_back( ( edge ){ f, 0, G[f].size() - 1 } ); 9. return ; 10. } 11. int dfs( int v, int t, int f ) { 12. if( v == t ) return f; 13. vis[v] = true; 14. for( int i = 0; i < G[v].size(); ++i ) { 15. edge &e = G[v][i]; 16. if( !vis[e.to] && e.cap > 0 ) { 17. int d = dfs( e.to, t, min( f, e.cap ) ); 18. if( d > 0 ) { 19. e.cap -= d; 20. G[e.to][e.rev] += d; 21. return d; 22. } 23. } 24. } 25. return 0; 26. } 27. int max\_flow( int s, int t ) { 28. int flow = 0; 29. while( true ) { 30. memset( vis, 0, sizeof( vis ) ); 31. int f = dfs( s, t, INF ); 32. if( f == 0 ) return flow; 33. flow += f; 34. } 35. return 0; 36. } 37. int main() { 38. int n, m; 39. int a, b, c; 40. scanf( "%d%d", &n, &m ); 41. for( int i = 0; i < n; ++i ) { 42. scanf( "%d%d%d", a, b, c ); 43. add( a, b, c ); 44. } 45. cout << max\_flow( 0, n - 1 ) << endl; 46. return 0; 47. } |

## 费用流

|  |
| --- |
| 1. typedef int MyType; 2. const MyType INF = 0x7F7F7F7F; 3. const int MAXN = 1000 + 10; 4. const int MAXE = 100000 + 10; 5. struct Edge { int to, next; MyType cap, cost; }; 6. Edge es[MAXE]; 7. int head[MAXN], que[MAXE], dis[MAXN], pre[MAXN]; 8. bool vis[MAXN]; 9. int n, m, cnt, src, des; 10. void add( int u, int v, MyType f, MyType c ) { 11. es[cnt].to = v; es[cnt].cap = f; es[cnt].cost = c; 12. es[cnt].next = head[u]; head[u] = cnt++; 13. es[cnt].to = u; es[cnt].cap = 0; es[cnt].cost = -c; 14. es[cnt].next = head[v]; head[v] = cnt++; 15. return ; 16. } 17. bool spfa() { 18. int mf, me; 19. memset( vis, false, sizeof( vis ) ); 20. memset( dis, 0x7F, sizeof( dis ) ); 21. memset( pre, -1, sizeof( pre ) ); 22. mf = me = 0; 23. que[me++] = src; dis[src] = 0; vis[src] = true; 24. while( mf < me ) { 25. int u = que[mf++]; 26. for( int i = head[u]; ~i; i = es[i].next ) { 27. int v = es[i].to; 28. if( es[i].cap > 0 && dis[v] > dis[u] + es[i].cost ) { 29. dis[v] = dis[u] + es[i].cost; 30. pre[v] = i; 31. if( !vis[v] ) { 32. vis[v] = true; 33. que[me++] = v; 34. } 35. } 36. } 37. vis[u] = false; 38. } 39. return dis[des] != INF; 40. } 41. MyType cflow() { 42. MyType flow = INF; 43. int u = des; 44. while( ~pre[u] ) { 45. u = pre[u]; 46. flow = min( flow, es[u].cap ); 47. u = es[u ^ 1].to; 48. } 49. u = des; 50. while( ~pre[u] ) { 51. u = pre[u]; 52. es[u].cap -= flow; 53. es[u ^ 1].cap += flow; 54. u = es[u ^ 1].to; 55. } 56. return flow; 57. } 58. MyType MCMF() { 59. MyType mincost, maxflow; 60. mincost = maxflow = 0; 61. while( spfa() ) { 62. MyType flow = cflow(); 63. maxflow += flow; 64. mincost += flow \* dis[des]; 65. } 66. return mincost; 67. } 68. int main() { 69. int a, b, c; 70. while( ~scanf( "%d%d", &n, &m ) ) { 71. memset( head, -1, sizeof( head ) ); 72. cnt = 0; 73. src = 0; des = n + 1; 74. add( src, 1, 2, 0 ); 75. for( int i = 0; i < m; ++i ) { 76. scanf( "%d%d%d", &a, &b, &c ); 77. add( a, b, 1, c ); 78. add( b, a, 1, c ); 79. } 80. add( n, des, 2, 0 ); 81. printf( "%d\n", MCMF() ); 82. } 83. return 0; 84. } |

## 欧拉路径

### Fleury

|  |
| --- |
| 1. int stk[1005]; 2. int top; 3. int N, M, ss, tt; 4. int mp[1005][1005]; 5. void dfs(int x) { 6. stk[top++] = x; 7. for (int i = 1; i <= N; ++i) { 8. if (mp[x][i]) { 9. mp[x][i] = mp[i][x] = 0; // 删除此边 10. dfs(i); 11. break; 12. } 13. } 14. } 15. void fleury(int ss) { 16. int brige; 17. top = 0; 18. stk[top++] = ss; // 将起点放入Euler路径中 19. while (top > 0) { 20. brige = 1; 21. for (int i = 1; i <= N; ++i) { // 试图搜索一条边不是割边（桥） 22. if (mp[stk[top-1]][i]) { 23. brige = 0; 24. break; 25. } 26. } 27. if (brige) { // 如果没有点可以扩展，输出并出栈 28. printf("%d ", stk[--top]); 29. } else { // 否则继续搜索欧拉路径 30. dfs(stk[--top]); 31. } 32. } 33. } 34. int main() { 35. int x, y, deg, num; 36. while (scanf("%d %d", &N, &M) != EOF) { 37. memset(mp, 0, sizeof (mp)); 38. for (int i = 0; i < M; ++i) { 39. scanf("%d %d", &x, &y); 40. mp[x][y] = mp[y][x] = 1; 41. } 42. for (int i = 1; i <= N; ++i) { 43. deg = num = 0; 44. for (int j = 1; j <= N; ++j) { 45. deg += mp[i][j]; 46. } 47. if (deg % 2 == 1) { 48. ss = i, ++num; 49. printf("%d\n", i); 50. } 51. } 52. if (num == 0 || num == 2) { 53. fleury(ss); 54. } else { 55. puts("No Euler path"); 56. } 57. } 58. return 0; 59. } |

## 全局最小割

|  |
| --- |
| 1. const int INF = 0x7F7F7F7F; 2. const int MAXN = 1000 + 10; 3. int mat[MAXN][MAXN], v[MAXN], dis[MAXN]; 4. bool vis[MAXN]; 5. int n, m; 6. int SW() { 7. int ret = INF; 8. for( int i = 0; i <= n; ++i ) v[i] = i; 9. while( n > 1 ) { 10. int pre = 0; 11. memset( vis, false, sizeof( vis ) ); 12. memset( dis, 0, sizeof( dis ) ); 13. for( int i = 1; i < n; ++i ) { 14. int k = -1; 15. for( int j = 1; j < n; ++j ) { 16. if( !vis[v[j]] ) { 17. dis[v[j]] += mat[v[pre]][v[j]]; 18. if( k == -1 || dis[v[k]] < dis[v[j]] ) k = j; 19. } 20. } 21. vis[v[k]] = true; 22. if( i == n - 1 ) { 23. ret = min( ret, dis[v[k]] ); 24. for( int j = 0; j < n; ++j ) { 25. mat[v[pre]][v[j]] = ( mat[v[j]][v[pre]] += mat[v[j]][v[k]] ); 26. } 27. v[k] = v[--n]; 28. } 29. pre = k; 30. } 31. } 32. return ret; 33. } 34. int main() { 35. int a, b, c; 36. while( ~scanf( "%d%d", &n, &m ) ) { 37. memset( mat, 0, sizeof( mat ) ); 38. while( m-- ) { 39. scanf( "%d%d%d", &a, &b, &c ); 40. mat[a][b] = ( mat[b][a] += c ); 41. } 42. printf( "%d\n", SW() ); 43. } 44. return 0; 45. } |

## 差分约束

|  |
| --- |
| 建图： 约束图  在一个差分约束系统Ax≤b中，m X n的线性规划矩阵A可被看做是n顶点，m条边的图的关联矩阵。对于i=1,2,…,n，图中的每一个顶点vi对应着n个未知量的一个xi。图中的每个有向边对应着关于两个未知量的m个不等式中的一个。  给定一个差分约束系统Ax≤b，相应的约束图是一个带权有向图G=(V,E)，其中V={v0,v1,…,vn}，而且E={ (vi,vj) : xj-xi≤bk是一个约束}∪{ (v0,v1) , (v0,v2) , … , (v0,vn) }。引入附加顶点v0是为了保证其他每个顶点均从v0可达。因此，顶点集合V由对应于每个未知量xi的顶点vi和附加的顶点v0组成。边的集合E由对应于每个差分约束条件的边与对应于每个未知量xi的边(v0,vi)构成。如果xj-xi≤bk是一个差分约束，则边(vi,vj)的权w(vi,vj)=bk（注意i和j不能颠倒），从v0出发的每条边的权值均为0。  定理：给定一差分约束系统Ax≤b，设G=(V,E)为其相应的约束图。如果G不包含负权回路，那么x=( d(v0,v1) , d(v0,v2) , … , d(v0,vn) )是此系统的一可行解，其中d(v0,vi)是约束图中v0到vi的最短路径(i=1,2,…,n)。如果G包含负权回路，那么此系统不存在可行解。  差分约束问题的求解  由上述定理可知，可以采用Bellman-Ford算法对差分约束问题求解。因为在约束图中，从源点v0到其他所有顶点间均存在边，因此约束图中任何负权回路均从v0可达。如果Bellman-Ford算法返回TRUE，则最短路径权给出了此系统的一个可行解；如果返回FALSE，则差分约束系统无可行解。  关于n个未知量m个约束条件的一个差分约束系统产生出一个具有n+1个顶点和n+m条边的约束图。因此采用Bellman-Ford算法，可以再O((n+1)(n+m))=O(n^2+nm)时间内将系统解决。此外，可以用SPFA算法进行优化，复杂度为O(km)，其中k 为常数。 |
| 1. const int INF = 0x7F7F7F7F; 2. const int MAXN = 100000 + 10; 3. const int MAXM = 1000000 + 10; 4. struct Edge { int to, cost, next; }; 5. Edge es[MAXM]; 6. int head[MAXN], dis[MAXN], sta[MAXM]; 7. bool vis[MAXN]; 8. int n, m, cnt; 9. void add( int u, int v, int w ) { 10. es[cnt].to = v; es[cnt].cost = w; es[cnt].next = head[u]; head[u] = cnt++; 11. return ; 12. } 13. void spfa() { 14. int top = 0; 15. memset( dis, 0x7F, sizeof( dis ) ); 16. memset( vis, false, sizeof( vis ) ); 17. dis[1] = 0; vis[1] = true; sta[top++] = 1; 18. while( top ) { 19. int u = sta[--top]; 20. for( int i = head[u]; ~i; i = es[i].next ) { 21. int v = es[i].to; 22. if( dis[v] > dis[u] + es[i].cost ) { 23. dis[v] = dis[u] + es[i].cost; 24. if( !vis[v] ) { 25. vis[v] = true; 26. sta[top++] = v; 27. } 28. } 29. } 30. vis[u] = false; 31. } 32. return ; 33. } 34. int main() { 35. int a, b, c; 36. memset( head, -1, sizeof( head ) ); 37. cnt = 0; 38. scanf( "%d%d", &n, &m ); 39. for( int i = 0; i < m; ++i ) { 40. scanf( "%d%d%d", &a, &b, &c ); 41. add( a, b, c ); 42. } 43. spfa(); 44. printf( "%d\n", dis[n] ); 45. return 0; 46. } |

# 数据结构

## LCA

|  |
| --- |
| HDU 2586 验过  1. 注意变量名不要和其他数据结构冲突  2. ST在线算法，需要大量预处理。RMQ会占用大量内存  3. head数组要清-1  4. query时需要确保前小后大  5. lca() 返回的是点编号 |
| 1. const int MAXN = 1e5 + 10; 2. const int POW = 32; 3. struct Edge { int to, next; }; 4. Edge es[MAXN << 1]; 5. int seq[MAXN << 1], dep[MAXN << 1], fir[MAXN], fa[MAXN]; 6. int lg2[MAXN << 1], dp[MAXN << 1][POW]; 7. int head[MAXN], a[MAXN]; 8. int n, m, cnt2, tot; 9. void add( int u, int v ) { es[cnt2].to = v; es[cnt2].next = head[u]; head[u] = cnt2++; } 10. void dfs( int u, int fu, int d ) { 11. seq[++tot] = u; dep[tot] = d; fir[u] = tot; fa[u] = fu; 12. update( 1, vec.size(), rt[u], rt[fa[u]], getid( a[u] ) ); 13. for( int i = head[u]; ~i; i = es[i].next ) { 14. int v = es[i].to; 15. if( v == fu ) continue; 16. dfs( v, u, d + 1 ); 17. seq[++tot] = u; dep[tot] = d; 18. } 19. } 20. void init\_lca() { 21. int tn = 2 \* n - 1; 22. lg2[0] = -1; for( int i = 1; i < ( MAXN << 1 ); ++i ) lg2[i] = ( ( i & ( i - 1 ) ) == 0 ) ? lg2[i - 1] + 1 : lg2[i - 1]; 23. for( int i = 1; i <= tn; ++i ) dp[i][0] = i; 24. for( int j = 1; j < 20; ++j ) { 25. for( int i = 1; i + ( 1 << j ) - 1 <= tn; ++i ) { 26. int a = dp[i][j - 1], b = dp[i + ( 1 << ( j - 1 ) )][j - 1]; 27. dp[i][j] = dep[a] < dep[b] ? a : b; 28. } 29. } 30. } 31. int lca( int x, int y ) { 32. int k = lg2[y - x + 1]; 33. int a = dp[x][k], b = dp[y - ( 1 << k ) + 1][k]; 34. return seq[dep[a] < dep[b] ? a : b]; 35. } 36. int main() { 37. memset( head, -1, sizeof head ); tot = cnt2 = 0; 38. scanf( "%d%d", &n, &m ); 39. for( int i = 1, u, v; i < n; ++i ) { scanf( "%d%d", &u, &v ); add( u, v ); add( v, u ); } 40. dfs( 1, 0, 1 ); 41. init\_lca(); 42. for( int i = 0, u, v; i < m; ++i ) { 43. scanf( "%d%d%d", &u, &v ); 44. if( fir[u] > fir[v] ) swap( u, v ); 45. // 46. printf( "%d\n", lca( fir[u], fir[v] ) ); 47. } 48. return 0; 49. } |

## RMQ

|  |
| --- |
| HDU 3183 已验  1. init()  i 从 1 开始  dp[i][0] 装最原始数据  min/max 比较具体情况具体分析，有时需要自己写，例如lca可能需要利用dep做比较  ！！！特别需要注意小于号还是小于等于号！！！  2. query()  x < y  min/max 同上 |
| 1. #include <bits/stdc++.h> 2. using namespace std; 3. const int MAXN = 1e5 + 10; 4. const int POW = 32; 5. int lg2[MAXN], dp[MAXN][POW], a[MAXN]; 6. int n; 7. void init( int tn ) { 8. lg2[0] = -1; for( int i = 1; i < MAXN; ++i ) lg2[i] = ( ( i & ( i - 1 ) ) == 0 ) ? lg2[i - 1] + 1 : lg2[i - 1]; 9. for( int i = 1; i <= tn; ++i ) dp[i][0] = i; 10. for( int j = 1; j <= 20; ++j ) { 11. for( int i = 1; i + ( 1 << j ) - 1 <= tn; ++i ) { 12. dp[i][j] = min( dp[i][j - 1], dp[i + ( 1 << ( j - 1 ) )][j - 1] ); 13. } 14. } 15. } 16. int query( int x, int y ) { 17. int k = lg2[y - x + 1]; 18. return min( dp[x][k], dp[y - ( 1 << k ) + 1][k] ); 19. } |

## 差分前缀和

|  |
| --- |
| 定义数组A，存在n个元素A[1]~A[n]  定义差分数组D，其中D[1] = A[1], D[i] = A[i] - A[i-1], D[n+1] = 0  观察可发现  性质(1)，即A[i] = Σ(D[1]…D[i])，证：  首先A[1] = D[1]，之后A[2] = D[1] + D[2] = A[1] + A[2] - A[1] = A[2]，利用归纳法可证  定义前缀和数组S，其中S[i] = Σ(A[1]…A[i])  观察可发现  性质：Σ(A[i]…A[j]) = S[j] - S[i-1] ，证：  A[j] = A[1] + … + A[i-1] + A[i] + … + A[j]  A[i-1] = A[1] + … + A[i-1]  定义某数组为X  X数组的前缀和数组为S  S的前缀和数组为SS  SS[i] = Σ(S[0]…S[i]) = i \* X[1] + (i-1) \* X[2] + … + 1 \* X[i] (从第1项开始每一项都有一个X[1]，从第2项开始每一项都有一个X[2])  = (i+1) \*(X[1] + X[2] + … + X[i]) - (X[1] + 2\* X[2] + … + i \* X[i])  令T数组为 T[i] = i \* X[i]，则SS[i] = (i+1) \* S[i] - T[i]  存在某数组X` ，其差分数组为X，那么上面的性质仍然成立。  由于X`数组的差分数组的前缀和就是它本身，所以SS和S数组对于X`数组都降了一级，SS变为了前缀和数组，而S变为了X`数组本身。  我们需要求出A数组的区间和，那么简单的方法就是利用A的前缀和数组S，S[j] - S[i-1] 即为 i…j区间的和。  假如我们因为某种原因，修改操作时，我们不操作数组A本身而操作数组A的差分数组，为了获得等价的效果，  计算区间和就表示为了关于差分数组的 SS[j] - SS[i-1]。 => 以上，查询操作得到了解答。  对于A数组和它的差分数组DS，我们令DS[i] += d，那么由于定义，  A[i-1] = DS[i-1] + A[i-2] 而 A[i] = DS[i] + A[i-1]  由此得出A[1…i-1]都没有发生变化，而A[i…end]由于递推都获得了d的增量。  为了构造一个获得增量的区间，例如i…j区间，我们按图索骥令DS[j+1] -= d  由此A[j+1…end]区间最终增加了 d - d 的增量，等于没修改。  于是我们得到修改区间i…j的方法， DS[i] += d 同时 DS[j+1] -= d。 => 以上，修改操作得到了解答。  捋顺数组之间的关系，我们回到SS[i] = (i+1) \* S[i] - T[i]中，  区间查询即为 SS[j] - SS[i-1]  区间修改即为 X[i] += d 同时 X[j+1] -= d  其中X是原数组的差分数组。  观察SS[i]的表达式，SS[i] = (i+1) \*(X[1] + X[2] + … + X[i]) - (X[1] + 2\* X[2] + … + i \* X[i])  该表达式可以表示为两个前缀和。  故求SS[j] - SS[i-1] 即转化为 查找 4 个前缀和 进行计算的过程（利用树状数组或者zkw线段树）  而全体过程完全不改变原数组X`，一切操作皆借助其差分数组X完成。 |

## 线段树

|  |
| --- |
| POJ 3468 验过  1. 有时候需要将节点改成结构体  2. dowork 有时候需要提前声明update函数并调用update |
| 1. typedef long long LL; 2. const int MAXN = 100000 + 10; 3. #define lson rt << 1, left, mid 4. #define rson rt << 1 | 1, mid + 1, right 5. LL num[MAXN << 2], lazy[MAXN << 2]; 6. int n, q; 7. void build( int rt, int left, int right ) { 8. lazy[rt] = 0; 9. if( left == right ) { scanf( "%I64d", &num[rt] ); return ; } 10. int mid = ( left + right ) >> 1; 11. build( lson ); 12. build( rson ); 13. num[rt] = num[rt << 1] + num[rt << 1 | 1]; 14. return ; 15. } 16. void dowork( int rt, int len ) { 17. if( lazy[rt] ) { 18. int son = rt << 1; num[son] += lazy[rt] \* ( len - ( len >> 1 ) ); lazy[son] += lazy[rt]; 19. son = rt << 1 | 1; num[son] += lazy[rt] \* ( len >> 1 ); lazy[son] += lazy[rt]; 20. lazy[rt] = 0; 21. } 22. return ; 23. } 24. void update( int rt, int left, int right, int l, int r, LL val ) { 25. if( left == l && right == r ) { 26. num[rt] += val \* ( r - l + 1 ); 27. if( l != r ) lazy[rt] += val; 28. return ; 29. } 30. dowork( rt, right - left + 1 ); 31. int mid = ( left + right ) >> 1; 32. if( r <= mid ) update( lson, l, r, val ); 33. else if( l > mid ) update( rson, l, r, val ); 34. else { 35. update( lson, l, mid, val ); 36. update( rson, mid + 1, r, val ); 37. } 38. num[rt] = num[rt << 1] + num[rt << 1 | 1]; 39. return ; 40. } 41. LL query( int rt, int left, int right, int l, int r ) { 42. if( left == l && right == r ) return num[rt]; 43. dowork( rt, right - left + 1 ); 44. int mid = ( left + right ) >> 1; 45. if( r <= mid ) return query( lson, l, r ); 46. else if( l > mid ) return query( rson, l, r ); 47. return query( lson, l, mid ) + query( rson, mid + 1, r ); 48. } 49. int main() { 50. int a, b, c; 51. char st[2]; 52. memset( num, 0, sizeof( num ) ); 53. scanf( "%d%d", &n, &q ); 54. build( 1, 1, n ); 55. while( q-- ) { 56. scanf( "%s", st ); 57. if( st[0] == 'Q' ) { 58. scanf( "%d%d", &a, &b ); 59. printf( "%I64d\n", query( 1, 1, n, a, b ) ); 60. } else { 61. scanf( "%d%d%d", &a, &b, &c ); 62. update( 1, 1, n, a, b, c ); 63. } 64. } 65. return 0; 66. } |

## 树状数组

|  |
| --- |
| 1. 下标从1开始  2. 注意x==0时的lowbit，有可能超时 |
| 1. const int MAXN = 1e5 + 10; 2. int c[MAXN], a[MAXN]; 3. int n; 4. int lowbit( int x ) { return x & -x; } 5. void add( int i, int x ) { 6. while( i <= n ) { 7. add[i] += x; 8. i += lowbit( i ); 9. } 10. } 11. int sum( int i ) { 12. int ret = 0; 13. while( i ) { 14. sum += c[i]; 15. i -= lowbit( i ); 16. } 17. return ret; 18. } |

## 可持久化数据结构

### 静态主席树

|  |
| --- |
| POJ 2104 验过  1. T大小需要计算，一般64倍足够  2. 每颗树的含义需要明确，此题中每棵树存的是离散化后数据域，  记录每个数据出现的次数，并非每个位置上是什么数。 |
| 1. const int MAXN = 1e5 + 10; 2. struct Node { int l, r, num; }; 3. Node T[MAXN << 6]; 4. vector<int> vec; 5. int root[MAXN], a[MAXN]; 6. int n, m, cnt; 7. int getid( int x ) { return upper\_bound( vec.begin(), vec.end(), x ) - vec.begin(); } 8. void update( int left, int right, int &x, int y, int pos ) { 9. T[++cnt] = T[y]; ++T[cnt].num; x = cnt; 10. if( left == right ) return ; 11. int mid = ( left + right ) >> 1; 12. if( mid >= pos ) update( left, mid, T[x].l, T[y].l, pos ); 13. else update( mid + 1, right, T[x].r, T[y].r, pos ); 14. return ; 15. } 16. int query( int left, int right, int x, int y, int k ) { 17. if( left == right ) return left; 18. int mid = ( left + right ) >> 1, ret = 0, sum = 0; 19. sum = T[T[y].l].num - T[T[x].l].num; 20. if( sum >= k ) ret = query( left, mid, T[x].l, T[y].l, k ); 21. else ret = query( mid + 1, right, T[x].r, T[y].r, k - sum ); 22. return ret; 23. } 24. int main() { 25. cnt = 0; 26. scanf( "%d%d", &n, &m ); 27. for( int i = 1, t; i <= n; ++i ) scanf( "%d", a + i ), vec.push\_back( a[i] ); 28. sort( vec.begin(), vec.end() ); vec.erase( unique( vec.begin(), vec.end() ), vec.end() ); 29. for( int i = 1; i <= n; ++i ) update( 1, vec.size(), root[i], root[i - 1], getid( a[i] ) ); 30. for( int i = 0, x, y, z; i < m; ++i ) { 31. scanf( "%d%d%d", &x, &y, &z ); 32. printf( "%d\n", vec[query( 1, vec.size(), root[x - 1], root[y], z ) - 1] ); 33. } 34. return 0; 35. } |

### 动态主席树

|  |
| --- |
| 没法验 ZOJ不开了  对于修改操作，只是修改M次，每次改变俩个值（减去原先的，加上现在的）  也就是说如果把所有初值都插入到树状数组里是不值得的，  所以我们分两部分来做，所有初值按照静态来建，内存O(nlogn)，  而修改部分保存在树状数组中，每次修改logn棵树，每次插入增加logn个节点  O(M\*logn\*logn+nlogn)  1. 一定要全部离线后再做其他操作！！包括修改的数据！！  2. 空间需要计算下  3. 树状数组前n项为修改位置对应数据；后n项为建树初始状态数据 |
| 1. const int MAXN = 1e5 + 10; 2. const int MAXM = 1e4 + 10; 3. struct Node { int ls, rs, sum; }; 4. Node tr[MAXN << 4]; 5. struct Opr { int flag, l, r, k; }; 6. Opr op[MAXM]; 7. vector<int> vec, q1, q2; 8. int a[MAXN], root[MAXN << 1]; 9. int n, m, tn, tot; 10. int getid( int x ) { return upper\_bound( vec.begin(), vec.end(), x ) - vec.begin(); } 11. inline int lowbit( int x ) { return x & -x; } 12. void build( int l, int r, int &x, int pos ) { 13. tr[++tot] = tr[x]; x = tot; ++tr[x].sum; 14. if( l == r ) return ; 15. int m = ( l + r ) >> 1; 16. if( pos <= m ) build( l, m, tr[x].ls, pos ); 17. else build( m + 1, r, tr[x].rs, pos ); 18. } 19. void insrt( int l, int r, int &x, int pos, int val ) { 20. if( x == 0 ) { tr[++tot] = tr[x]; x = tot; } 21. tr[x].sum += val; 22. if( l == r ) return ; 23. int m = ( l + r ) >> 1; 24. if( pos <= m ) insrt( l, m, tr[x].ls, pos, val ); 25. else insrt( m + 1, r, tr[x].rs, pos, val ); 26. } 27. void bitinsrt( int pos, int x, int val ) { 28. int t = getid( x ); 29. for( int i = pos; i <= n; i += lowbit( i ) ) insrt( 1, tn, root[i], t, val ); 30. } 31. int qury( int l, int r, vector<int> &q1, vector<int> &q2, int k ) { 32. if( l == r ) return l; 33. int cnt = 0, m = ( l + r ) >> 1; 34. for( int i = 0; i < q1.size(); ++i ) cnt -= tr[tr[q1[i]].ls].sum; 35. for( int i = 0; i < q2.size(); ++i ) cnt += tr[tr[q2[i]].ls].sum; 36. for( int i = 0; i < q1.size(); ++i ) q1[i] = ( cnt >= k ? tr[q1[i]].ls : tr[q1[i]].rs ); 37. for( int i = 0; i < q2.size(); ++i ) q2[i] = ( cnt >= k ? tr[q2[i]].ls : tr[q2[i]].rs ); 38. if( cnt >= k ) return qury( l, m, q1, q2, k ); 39. else return qury( m + 1, r, q1, q2, k - cnt ); 40. } 41. int bitqury( int l, int r, int k ) { 42. q1.clear(); q2.clear(); 43. q1.push\_back( root[l == 1 ? 0 : l - 1 + n] ); 44. q2.push\_back( root[r + n] ); 45. for( int i = l - 1; i > 0; i -= lowbit( i ) ) q1.push\_back( root[i] ); 46. for( int i = r; i > 0; i -= lowbit( i ) ) q2.push\_back( root[i] ); 47. return vec[qury( 1, tn, q1, q2, k ) - 1]; 48. } 49. int main() { 50. char s[2]; 51. int t; 52. scanf( "%d", &t ); 53. while( t-- ) { 54. vec.clear(); tot = 0; 55. memset( root, 0, sizeof root ); 56. scanf( "%d%d", &n, &m ); 57. for( int i = 1; i <= n; ++i ) scanf( "%d", a + i ), vec.push\_back( a[i] ); 58. for( int i = 0, x, y, z; i < m; ++i ) { 59. scanf( "%s", s ); 60. if( s[0] == 'Q' ) { 61. op[i].flag = 0; 62. scanf( "%d%d%d", &op[i].l, &op[i].r, &op[i].k ); 63. } else { 64. op[i].flag = 1; 65. scanf( "%d%d", &op[i].l, &op[i].r ); 66. vec.push\_back( op[i].r ); 67. } 68. } 69. sort( vec.begin(), vec.end() ); tn = unique( vec.begin(), vec.end() ) - vec.begin(); 70. for( int i = 1; i <= n; ++i ) { 71. root[i + n] = root[i - 1 + n]; 72. build( 1, tn, root[i + n], getid( a[i] ) ); 73. } 74. for( int i = 0; i < m; ++i ) { 75. if( op[i].flag == 0 ) printf( "%d\n", bitqury( op[i].l, op[i].r, op[i].k ) ); 76. else { 77. bitinsrt( op[i].l, a[op[i].l], -1 ); 78. bitinsrt( op[i].l, op[i].r, 1 ); 79. a[op[i].l] = op[i].r; 80. } 81. } 82. } 83. return 0; 84. } |

### 可持久化线段树

|  |
| --- |
| HDU 4348 已验  lazy 不下放，只标记，查询时带上即可 省去许多因下放而产生的新节点 |
| 1. typedef long long LL; 2. const int MAXN = 1e5 + 10; 3. struct Node { int ls, rs, add; LL sum; }; 4. Node tr[MAXN << 5]; 5. int root[MAXN]; 6. int n, m, tot; 7. int build( int l, int r ) { 8. int cur = ++tot; 9. tr[cur].add = 0; 10. if( l == r ) { scanf( "%I64d", &tr[cur].sum ); return cur; } 11. int mid = ( l + r ) >> 1; 12. tr[cur].ls = build( l, mid ); 13. tr[cur].rs = build( mid + 1, r ); 14. tr[cur].sum = tr[tr[cur].ls].sum + tr[tr[cur].rs].sum; 15. return cur; 16. } 17. void update( int l, int r, int left, int right, int &x, int y, int val ) { 18. x = ++tot; tr[x] = tr[y]; 19. tr[x].sum += val \* ( right - left + 1 ); 20. if( left <= l && r <= right ) { 21. tr[x].add += val; 22. return ; 23. } 24. int mid = ( l + r ) >> 1; 25. if( right <= mid ) update( l, mid, left, right, tr[x].ls, tr[y].ls, val ); 26. else if( mid < left ) update( mid + 1, r, left, right, tr[x].rs, tr[y].rs, val ); 27. else { 28. update( l, mid, left, mid, tr[x].ls, tr[y].ls, val ); 29. update( mid + 1, r, mid + 1, right, tr[x].rs, tr[y].rs, val ); 30. } 31. } 32. LL query( int l, int r, int left, int right, int rt ) { 33. if( left <= l && r <= right ) return tr[rt].sum; 34. LL ret = tr[rt].add \* ( right - left + 1 ); 35. int mid = ( l + r ) >> 1; 36. if( right <= mid ) return ret + query( l, mid, left, right, tr[rt].ls ); 37. else if( mid < left ) return ret + query( mid + 1, r, left, right, tr[rt].rs ); 38. else return ret + query( l, mid, left, mid, tr[rt].ls ) + query( mid + 1, r, mid + 1, right, tr[rt].rs ); 39. } 40. int main() { 41. char s[5]; 42. int cur; 43. while( ~scanf( "%d%d", &n, &m ) ) { 44. tot = 0; 45. root[cur = 0] = build( 1, n ); 46. for( int i = 0, x, y, z; i < m; ++i ) { 47. scanf( "%s", s ); 48. if( s[0] == 'C' ) { 49. scanf( "%d%d%d", &x, &y, &z ); ++cur; 50. update( 1, n, x, y, root[cur], root[cur - 1], z ); 51. } else if( s[0] == 'Q' ) { 52. scanf( "%d%d", &x, &y ); 53. printf( "%I64d\n", query( 1, n, x, y, root[cur] ) ); 54. } else if( s[0] == 'H' ) { 55. scanf( "%d%d%d", &x, &y, &z ); 56. printf( "%I64d\n", query( 1, n, x, y, root[z] ) ); 57. } else scanf( "%d", &cur ); 58. } 59. } 60. return 0; 61. } |

## 划分树

|  |
| --- |
| HDU 3473 已验  求区间第K大  cnt数组 区间内有多少 <= 中位数的数字  num数组 30层划分树，存原始数组及排序后的数组  sum数组 前缀和  leftsum 区间小于中位数的数前缀和 |
| 1. typedef long long LL; 2. const int MAXN = 1e5 + 10; 3. LL sum[MAXN], leftsum[30][MAXN]; 4. int a[MAXN]; 5. int num[30][MAXN], cnt[30][MAXN]; 6. int n, q; 7. LL lsum, rsum, lcnt, rcnt; 8. void build( int left, int right, int dep ) { 9. if( left == right ) return ; 10. int mid = ( left + right ) >> 1, ncnt = mid - left + 1; 11. for( int i = left; i <= right; ++i ) if( num[dep][i] < a[mid] ) --ncnt; 12. int lp = left, rp = mid + 1; 13. for( int i = left; i <= right; ++i ) { 14. if( i == left ) cnt[dep][i] = 0; 15. else cnt[dep][i] = cnt[dep][i - 1]; 16. if( num[dep][i] < a[mid] ) { 17. ++cnt[dep][i]; 18. num[dep + 1][lp++] = num[dep][i]; 19. leftsum[dep][i] = leftsum[dep][i - 1] + num[dep][i]; 20. } else if( num[dep][i] > a[mid] ) { 21. num[dep + 1][rp++] = num[dep][i]; 22. leftsum[dep][i] = leftsum[dep][i - 1]; 23. } else { 24. if( ncnt ) { 25. --ncnt; ++cnt[dep][i]; 26. num[dep + 1][lp++] = num[dep][i]; 27. leftsum[dep][i] = leftsum[dep][i - 1] + num[dep][i]; 28. } else { 29. num[dep + 1][rp++] = num[dep][i]; 30. leftsum[dep][i] = leftsum[dep][i - 1]; 31. } 32. } 33. } 34. build( left, mid, dep + 1 ); 35. build( mid + 1, right, dep + 1 ); 36. return ; 37. } 38. int query( int l, int r, int k, int left, int right, int dep ) { 39. if( l == r ) return num[dep][l]; 40. int mid = ( left + right ) >> 1, s, ss, tmp; 41. if( l == left ) { s = cnt[dep][r]; ss = 0; } 42. else { s = cnt[dep][r] - cnt[dep][l - 1]; ss = cnt[dep][l - 1]; } 43. if( s >= k ) { 44. l = left + ss; 45. r = left + ss + s - 1; 46. tmp = query( l, r, k, left, mid, dep + 1 ); 47. } else { 48. lcnt += s; 49. lsum += leftsum[dep][r] - leftsum[dep][l - 1]; 50. l = mid + 1 + l - left - ss; 51. r = mid + 1 + r - left - cnt[dep][r]; 52. tmp = query( l, r, k - s, mid + 1, right, dep + 1 ); 53. } 54. return tmp; 55. } 56. int main() { 57. int t, x, y, mid, tt = 0; 58. LL k; 59. scanf( "%d", &t ); 60. while( t-- ) { 61. printf( "Case #%d:\n", ++tt ); 62. sum[0] = 0; 63. scanf( "%d", &n ); 64. for( int i = 1; i <= n; ++i ) { 65. scanf( "%d", a + i ); 66. num[0][i] = a[i]; 67. sum[i] = sum[i - 1] + a[i]; 68. } 69. sort( a + 1, a + 1 + n ); 70. build( 1, n, 0 ); 71. scanf( "%d", &q ); 72. while( q-- ) { 73. scanf( "%d%d", &x, &y ); ++x; ++y; 74. lsum = lcnt = 0; 75. mid = query( x, y, ( y - x ) / 2 + 1, 1, n, 0 ); 76. rcnt = y - x + 1 - lcnt; 77. rsum = sum[y] - sum[x - 1] - lsum; 78. k = rsum - lsum + mid \* ( lcnt - rcnt ); 79. printf( "%I64d\n", k ); 80. } 81. puts( "" ); 82. } 83. return 0; 84. } |

## 莫队算法

### 莫队分块

|  |
| --- |
| HYSBZ 2038 已验  这是离线算法，需要离散化数据  1. update函数要求O(1)转移  2. update的第二个参数依据具体情况定  3. 间隔dm一般取sqrt(n)，特殊情况卡空间可以改成n^(2/3)  4. 下标从1开始 |
| 1. typedef long long LL; 2. const int MAXN = 100000 + 10; 3. struct Node { int l, r, id; LL a, b; }; 4. Node node[MAXN]; 5. LL cnt[MAXN], ans; 6. int c[MAXN], pos[MAXN]; 7. int n, m; 8. LL gcd( LL a, LL b ) { return b ? gcd( b, a % b ) : a; } 9. bool cmp( const Node &a, const Node &b ) { 10. return pos[a.l] == pos[b.l] ? a.r < b.r : a.l < b.l; 11. } 12. bool cmp\_id( const Node &a, const Node &b ) { return a.id < b.id; } 13. // O(1) 14. void update( int x, int d ) { 15. ans -= cnt[c[x]] \* cnt[c[x]]; 16. cnt[c[x]] += d; 17. ans += cnt[c[x]] \* cnt[c[x]]; 18. return ; 19. } 20. void solve() { 21. ans = 0; 22. for( int i = 0, l = 1, r = 0; i < m; ++i ) { 23. while( r < node[i].r ) update( ++r, 1 ); 24. while( r > node[i].r ) update( r--, -1 ); 25. while( l < node[i].l ) update( l++, -1 ); 26. while( l > node[i].l ) update( --l, 1 ); 27. // solve 28. node[i].a = ans - ( r - l + 1 ); 29. node[i].b = 1LL \* ( r - l + 1 ) \* ( r - l ); 30. LL k = gcd( node[i].a, node[i].b ); 31. node[i].a /= k; node[i].b /= k; 32. } 33. return ; 34. } 35. int main() { 36. scanf( "%d%d", &n, &m ); 37. int dm = ( int )sqrt( n ); 38. for( int i = 1; i <= n; ++i ) { 39. scanf( "%d", c + i ); 40. pos[i] = ( i - 1 ) / dm + 1; 41. } 42. for( int i = 0; i < m; ++i ) { 43. scanf( "%d%d", &node[i].l, &node[i].r ); 44. node[i].id = i; 45. } 46. sort( node, node + m, cmp ); 47. solve(); 48. sort( node, node + m, cmp\_id ); 49. for( int i = 0; i < m; ++i ) printf( "%lld/%lld\n", node[i].a, node[i].b ); 50. return 0; 51. } |

### 树上莫队

|  |
| --- |
| A. 子树树上莫队  现在有一棵树，有n个节点，节点有点权，每次询问一个子树内的不重复数个数。  1<=n,q<=10^5，1<=点权<=10^9。  这个题显然比较trivial嘛...先把点权离散一下，然后一遍dfs搞出dfs序，那么一个子树就对应dfs序上一段，所以我们就可以在dfs序上莫队，开一个数组记一下每个数的出现次数。  B. 路径树上莫队  现在有一棵树，有n个节点，节点有点权，每次询问一条路径上的不重复数个数。  1<=n,q<=10^5，1<=点权<=10^9。  莫队用不了了？我们重新定义一个dfs序！  我们在开始访问和结束访问一个点的时候都记一下时间戳，我们设开始访问的时间为st，结束访问的时间为ed。  我们假设要询问一条路径a-b，设lca为p=lca(a,b)。不妨设st[a]<=st[b]（否则交换一下）。  当p=a时，这应该是一个比较简单的情形：a-b是一段父子链。  我们考虑这个新dfs序上[st[a],st[b]]的点，我们可以发现，a-b上的点被算了一遍，其他点都被算了2遍或0遍！那么我们统计的时候注意一下就可以了。  当p≠a时，我们也要一样统计[ed[a],st[b]]的点（从ed[a]开始为保证a不会被排除掉），但是这回lca会被重复统计，所以要另外算一下。 |

## 树链剖分

|  |
| --- |
| 还在学习姿势ing |
| 1. typedef long long LL; 2. const int MAXN = 10000 + 10; 3. const int MAXM = 1000000 + 10; 4. #define MID(x, y) (((x) + (y)) >> 1) 5. int fa[MAXN], top[MAXN], w[MAXN], son[MAXN], dep[MAXN], sz[MAXN], r[MAXN]; 6. int a[MAXN], b[MAXN]; 7. LL c[MAXN]; 8. int ind[MAXN]; 9. int t[MAXM], nt[MAXM]; 10. int cnt1, cnt2, cnt3; 11. int n, m; 12. struct node { 13. int l, r; 14. int a, b; 15. LL sum; 16. }f[MAXM]; 17. int rt; 18. void dfs1( int x, int d ) { 19. dep[x] = d; son[x] = 0; sz[x] = 1; 20. for( int k = ind[x]; ~k; k = nt[k] ) { 21. if( t[k] != fa[x] ) { 22. fa[t[k]] = x; 23. dfs1( t[k], d + 1 ); 24. sz[x] += sz[t[k]]; 25. if( sz[t[k]] > sz[son[x]] ) son[x] = t[k]; 26. } 27. } 28. return ; 29. } 30. void dfs2( int x, int tt ) { 31. w[x] = ++cnt2; top[x] = tt; 32. if( son[x] ) dfs2( son[x], tt ); 33. for( int k = ind[x]; ~k; k = nt[k] ) { 34. if( t[k] != fa[x] && t[k] != son[x] ) 35. dfs2( t[k], t[k] ); 36. } 37. return ; 38. } 39. void add( int a, int b ) { 40. t[cnt1] = b; nt[cnt1] = ind[a]; ind[a] = cnt1++; 41. return ; 42. } 43. void update( int x ) { 44. f[x].sum = f[f[x].l].sum + f[f[x].r].sum; 45. } 46. int bt( int a, int b ) { 47. int x = cnt3++; 48. f[x].a = a; f[x].b = b; 49. if( a < b ) { 50. int mid = MID( a, b ); 51. f[x].l = bt( a, mid ); 52. f[x].r = bt( mid + 1, b ); 53. f[x].sum = 0; 54. } else f[x].sum = 0; 55. return x; 56. } 57. // Query On ST, Do not Call Directly 58. LL query( int x, int a, int b ) { 59. if( a <= f[x].a && f[x].b <= b ) return f[x].sum; 60. int mid = MID( f[x].a, f[x].b ); 61. LL ans = 0; 62. if( a <= mid ) ans += query( f[x].l, a, b ); 63. if( b > mid ) ans += query( f[x].r, a, b ); 64. return ans; 65. } 66. //Modify Point 67. void update( int x, int p, int cc ) { 68. if( f[x].a == f[x].b ) { f[x].sum = cc; return; } 69. int mid = MID( f[x].a, f[x].b ); 70. if( p <= mid ) update( f[x].l, p, cc ); 71. else update( f[x].r, p, cc ); 72. update( x ); 73. return ; 74. } 75. //Query Segment 76. LL query( int x, int y ) { 77. int fx = top[x], fy = top[y]; 78. LL sum = 0; 79. while( fx != fy ) { 80. if( dep[fx] < dep[fy] ) { 81. swap( x, y ); 82. swap( fx, fy ); 83. } 84. sum += query( rt, w[fx], w[x] ); 85. x = fa[top[x]]; 86. fx = top[x]; 87. } 88. if( dep[x] > dep[y] ) swap( x, y ); 89. if( x == y ) return sum; 90. return sum + query( rt, w[son[x]], w[y] ); 91. } 92. int main() { 93. return 0; 94. } |

# 动态规划

## 树DP

### 经典

|  |
| --- |
| 1. const int MAXN = 1e5 + 10; 2. struct Edge { int to, cost, next; }; 3. Edge es[MAXN << 2]; 4. int head[MAXN], nmax[MAXN][2], son[MAXN][2]; 5. int n, cnt; 6. void add( int u, int v, int w ) { 7. es[cnt].to = v; es[cnt].cost = w; es[cnt].next = head[u]; head[u] = cnt++; 8. return ; 9. } 10. void dfs1( int u, int fa ) { 11. for( int i = head[u]; ~i; i = es[i].next ) { 12. int v = es[i].to; 13. if( v == fa ) continue; 14. dfs1( v, u ); 15. int tmp = nmax[v][0] + es[i].cost; 16. if( tmp > nmax[u][0] ) { 17. nmax[u][1] = nmax[u][0]; son[u][1] = son[u][0]; 18. nmax[u][0] = tmp; son[u][0] = v; 19. } else if( tmp > nmax[u][1] ) { 20. nmax[u][1] = tmp; son[u][1] = v; 21. } 22. } 23. return ; 24. } 25. void dfs2( int u, int fa, int len ) { 26. int tmp; 27. if( son[fa][0] != u ) tmp = len + nmax[fa][0]; 28. else tmp = len + nmax[fa][1]; 29. if( tmp > nmax[u][0] ) { 30. nmax[u][1] = nmax[u][0]; son[u][1] = son[u][0]; 31. nmax[u][0] = tmp; son[u][0] = fa; 32. } else if( tmp > nmax[u][1] ) { 33. nmax[u][1] = tmp; son[u][1] = fa; 34. } 35. for( int i = head[u]; ~i; i = es[i].next ) { 36. int v = es[i].to; 37. if( v == fa ) continue; 38. dfs2( v, u, es[i].cost ); 39. } 40. return ; 41. } 42. int main() { 43. int ta, tb; 44. while( ~scanf( "%d", &n ) ) { 45. memset( head, -1, sizeof( head ) ); 46. memset( nmax, 0, sizeof( nmax ) ); 47. memset( son, -1, sizeof( son ) ); 48. for( int i = 2; i <= n; ++i ) { 49. scanf( "%d%d", &ta, &tb ); 50. add( i, ta, tb ); 51. add( ta, i, tb ); 52. } 53. dfs1( 1, 0 ); 54. dfs2( 1, 0, 0 ); 55. for( int i = 1; i <= n; ++i ) printf( "%d\n", nmax[i][0] ); 56. } 57. return 0; 58. } |

### 删点

|  |
| --- |
| 1. const int INF = 0x3F3F3F3F; 2. const int MAXN = 1e5 + 10; 3. struct Edge { int to, next; }; 4. Edge es[MAXN << 2]; 5. int head[MAXN], son[MAXN], dp[MAXN], ans[MAXN]; 6. int n, m, cnt, nmin; 7. void add( int u, int v ) { 8. es[cnt].to = v; es[cnt].next = head[u]; head[u] = cnt++; 9. return ; 10. } 11. void dfs( int u, int fa ) { 12. son[u] = 1; dp[u] = 0; 13. for( int i = head[u]; ~i; i = es[i].next ) { 14. int v = es[i].to; 15. if( v == fa ) continue; 16. dfs( v, u ); 17. son[u] += son[v]; 18. dp[u] = max( dp[u], son[v] ); 19. } 20. dp[u] = max( dp[u], n - son[u] ); 21. nmin = min( nmin, dp[u] ); 22. return ; 23. } 24. int main() { 25. int ta, tb; 26. while( ~scanf( "%d", &n ) ) { 27. memset( head, -1, sizeof( head ) ); 28. cnt = 0; nmin = INF; 29. for( int i = 1; i < n; ++i ) { 30. scanf( "%d%d", &ta, &tb ); 31. add( ta, tb ); add( tb, ta ); 32. } 33. dfs( 1, -1 ); 34. int tn = 0; 35. for( int i = 1; i <= n; ++i ) { 36. if( dp[i] == nmin ) ans[tn++] = i; 37. } 38. for( int i = 0; i < tn - 1; ++i ) printf( "%d ", ans[i] ); 39. printf( "%d\n", ans[tn - 1] ); 40. } 41. return 0; 42. } |

### 树上背包

|  |
| --- |
| 1. const int NINF = 0x80808080; 2. const int MAXN = 3e3 + 10; 3. struct Edge { int to, cost, next; }; 4. Edge es[MAXN << 2]; 5. int head[MAXN], dp[MAXN][MAXN], val[MAXN], num[MAXN]; 6. int n, m, tn, cnt; 7. void add( int u, int v, int w ) { 8. es[cnt].to = v; es[cnt].cost = w; es[cnt].next = head[u]; head[u] = cnt++; 9. return ; 10. } 11. void dfs( int u, int fa ) { 12. memset( dp[u], 0x80, sizeof( dp[u] ) ); dp[u][0] = 0; 13. if( head[u] == -1 ) { dp[u][1] = val[u]; return ; } 14. for( int i = head[u]; ~i; i = es[i].next ) { 15. int v = es[i].to, cost = es[i].cost; 16. if( v == fa ) continue; 17. dfs( v, u ); 18. num[u] += num[v]; 19. for( int j = num[u]; j >= 1; --j ) { 20. for( int k = 0; k < j; ++k ) 21. dp[u][j] = max( dp[u][j], dp[u][k] + dp[v][j - k] - cost ); 22. } 23. } 24. return ; 25. } 26. int main() { 27. int ta, tb; 28. while( ~scanf( "%d%d", &n, &m ) ) { 29. memset( head, -1, sizeof( head ) ); 30. memset( num, 0, sizeof( num ) ); 31. cnt = 0; 32. for( int i = 1; i <= n - m; ++i ) { 33. scanf( "%d", &tn ); 34. for( int j = 0; j < tn; ++j ) { 35. scanf( "%d%d", &ta, &tb ); 36. add( i, ta, tb ); 37. } 38. val[i] = 0; 39. } 40. for( int i = n - m + 1; i <= n; ++i ) { scanf( "%d", val + i ); num[i] = 1; } 41. dfs( 1, -1 ); 42. for( int i = num[1]; i >= 0; --i ) { 43. if( dp[1][i] >= 0 ) { 44. printf( "%d\n", i ); 45. break; 46. } 47. } 48. } 49. return 0; 50. } |

### 应用：树上任意点能到达的最远距离

|  |
| --- |
| 1. #define INF 0x3f3f3f3f 2. const int MAXN = 50000 + 10; 3. const int MAXM = 100000 + 10; 4. struct Edge { int v, w, next; }; 5. Edge es[MAXM]; 6. int head[MAXN], mmax[MAXN][2], poi[MAXN][2], dis[MAXN], disf[MAXN][2], mlog[MAXN]; 7. int d1[MAXN][17], d2[MAXN][17]; 8. int n, m, cnt; 9. void add( int u, int v, int w ) { 10. es[cnt].v = v; es[cnt].w = w; es[cnt].next = head[u]; head[u] = cnt++; 11. return ; 12. } 13. void dfs( int u, int pre ) { 14. bool flag = false; 15. for( int i = head[u]; ~i; i = es[i].next ) { 16. int v = es[i].v, w = es[i].w; 17. if( v != pre ) { 18. flag = true; 19. dfs( v, u ); 20. if( mmax[u][0] < mmax[v][0] + w ) { 21. mmax[u][1] = mmax[u][0]; 22. mmax[u][0] = mmax[v][0] + w; 23. poi[u][1] = poi[u][0]; 24. poi[u][0] = v; 25. } else if( mmax[u][1] < mmax[v][0] + w ) { 26. mmax[u][1] = mmax[v][0] + w; 27. poi[u][1] = v; 28. } 29. } 30. } 31. if( !flag ) { 32. dis[u] = 0; 33. mmax[u][0] = mmax[u][1] = 0; 34. poi[u][0] = poi[u][1] = 0; 35. } 36. return ; 37. } 38. void dfs2( int u, int pre ) { 39. for( int i = head[u]; ~i; i = es[i].next ) { 40. int v = es[i].v, w = es[i].w; 41. if( v != pre ) { 42. if( v == poi[u][0] ) { 43. dis[v] = max( mmax[v][0], w + disf[u][0] ); 44. disf[v][0] = max( disf[u][0] + w, mmax[v][1] ); 45. disf[v][1] = max( disf[u][0] + w, mmax[v][0] ); 46. } else { 47. dis[v] = max( mmax[v][0], w + disf[u][1] ); 48. disf[v][0] = max( disf[u][1] + w, mmax[v][1] ); 49. disf[v][1] = max( disf[u][1] + w, mmax[v][0] ); 50. } 51. dfs2( v, u ); 52. } 53. } 54. return ; 55. } 56. int main() { 57. int a, b, c, q; 58. while( ~scanf( "%d%d", &n, &m ) && n + m ) { 59. memset( head, -1, sizeof( head ) ); 60. memset( mmax, 0, sizeof( mmax ) ); 61. memset( dis, 0, sizeof( dis ) ); 62. cnt = 0; 63. add( 0, 1, 0 ); 64. for( int i = 0; i < n - 1; ++i ) { 65. scanf( "%d%d%d", &a, &b, &c ); 66. add( a, b, c ); add( b, a, c ); 67. } 68. dfs( 0, 0 ); 69. disf[0][0] = disf[0][1] = 0; 70. poi[0][0] = 1; poi[0][1] = 0; 71. dfs2( 0, 0 ); 72. } 73. return 0; 74. } |

## 区间DP

|  |
| --- |
| const int INF = 0x3F3F3F3F;  const int MAXN = 100 + 10;  int a[MAXN], dp[MAXN][MAXN];  int n;  int main() {  memset( dp, 0, sizeof( dp ) );  scanf( "%d", &n );  for( int i = 1; i <= n; ++i ) scanf( "%d", a + i );  for( int m = 3; m <= n; ++m ) {  for( int i = 1; i <= n - m + 1; ++i ) {  int j = i + m - 1;  dp[i][j] = INF;  for( int k = i + 1; k < j; ++k ) {  dp[i][j] = min( dp[i][j], dp[i][k] + dp[k][j] + a[i] \* a[k] \* a[j] );  }  }  }  printf( "%d\n", dp[1][n] );  return 0;  } |

## 数位DP

|  |
| --- |
| 1. typedef long long LL; 2. const int MOD = 2520; 3. LL dp[21][MOD + 10][50]; 4. int index[MOD + 10], dig[21]; 5. void init() { 6. int cnt = 0; 7. for( int i = 1; i <= MOD; ++i ) if( MOD % i == 0 ) index[i] = cnt++; 8. memset( dp, -1, sizeof( dp ) ); 9. return ; 10. } 11. int gcd( int a, int b ) { return b == 0 ? a : gcd( b, a % b ); } 12. int lcm( int a, int b ) { return a / gcd( a, b ) \* b; } 13. LL dfs( int pos, int presum, int prelcm, bool edge ) { 14. if( pos == -1 ) return presum % prelcm == 0; 15. if( !edge && dp[pos][presum][index[prelcm]] != -1 ) { 16. return dp[pos][presum][index[prelcm]]; 17. } 18. LL ret = 0; 19. int ed = edge ? dig[pos] : 9; 20. for( int i = 0; i <= ed; ++i ) { 21. int nowsum = ( presum \* 10 + i ) % MOD; 22. int nowlcm = prelcm; 23. if( i ) nowlcm = lcm( prelcm, i ); 24. ret += dfs( pos - 1, nowsum, nowlcm, edge && ( i == ed ) ); 25. } 26. if( !edge ) dp[pos][presum][index[prelcm]] = ret; 27. return ret; 28. } 29. LL gao( LL x ) { 30. int pos = 0; 31. while( x ) { 32. dig[pos++] = x % 10; 33. x /= 10; 34. } 35. return dfs( pos - 1, 0, 1, true ); 36. } 37. int main() { 38. int t; 39. init(); 40. scanf( "%d", &t ); 41. while( t-- ) { 42. LL l, r; 43. scanf( "%I64d %I64d", &l, &r ); 44. printf( "%I64d\n", gao( r ) - gao( l - 1 ) ); 45. } 46. return 0; 47. } |

## 可能的树分治模板

|  |
| --- |
| 1. const int INF = 0x3F3F3F3F; 2. const int MAXN = 1e4 + 10; 3. struct Edge { int to, cost, next; }; 4. Edge es[MAXN << 1]; 5. int head[MAXN], son[MAXN], dp[MAXN], dep[MAXN]; 6. bool vis[MAXN]; 7. vector<int> vdep; 8. int n, m, cnt, root, size, ans; 9. void add( int u, int v, int w ) { 10. es[cnt].to = v; es[cnt].cost = w; es[cnt].next = head[u]; head[u] = cnt++; 11. return ; 12. } 13. void getroot( int u, int fa ) { 14. son[u] = 1; dp[u] = 0; 15. for( int i = head[u]; ~i; i = es[i].next ) { 16. int v = es[i].to; 17. if( v == fa || vis[v] ) continue; 18. getroot( v, u ); 19. son[u] += son[v]; 20. dp[u] = max( dp[u], son[v] ); 21. } 22. dp[u] = max( dp[u], size - son[u] ); 23. if( dp[u] < dp[root] ) root = u; 24. return ; 25. } 26. void getdep( int u, int fa ) { 27. vdep.push\_back( dep[u] ); 28. son[u] = 1; 29. for( int i = head[u]; ~i; i = es[i].next ) { 30. int v = es[i].to; 31. if( v == fa || vis[v] ) continue; 32. dep[v] = dep[u] + es[i].cost; 33. getdep( v, u ); 34. son[u] += son[v]; 35. } 36. return ; 37. } 38. int calc( int u, int init ) { 39. int ret = 0; 40. vdep.clear(); dep[u] = init; 41. getdep( u, 0 ); 42. sort( vdep.begin(), vdep.end() ); 43. int l = 0, r = vdep.size() - 1; 44. while( l < r ) { 45. if( vdep[l] + vdep[r] <= m ) { ret += r - l; ++l; } 46. else r--; 47. } 48. return ret; 49. } 50. void solve( int u ) { 51. ans += calc( u, 0 ); 52. vis[u] = true; 53. for( int i = head[u]; ~i; i = es[i].next ) { 54. int v = es[i].to; 55. if( vis[v] ) continue; 56. ans -= calc( v, es[i].cost ); 57. dp[0] = size = son[v]; 58. getroot( v, root = 0 ); 59. solve( root ); 60. } 61. return ; 62. } 63. int main() { 64. int ta, tb, tc; 65. while( ~scanf( "%d%d", &n, &m ) && n + m ) { 66. memset( vis, false, sizeof( vis ) ); 67. memset( head, -1, sizeof( head ) ); 68. cnt = 0; 69. for( int i = 1; i < n; ++i ) { 70. scanf( "%d%d%d", &ta, &tb, &tc ); 71. add( ta, tb, tc ); add( tb, ta, tc ); 72. } 73. ans = 0; dp[0] = size = n; 74. getroot( 1, root = 0 ); 75. solve( root ); 76. printf( "%d\n", ans ); 77. } 78. return 0; 79. } |

# 字符串

## AC自动机

|  |
| --- |
| HDU 2222 验过  1. 数组用法  len 长度 end 模式串终结符  依据需要添加数组记录相应信息  2. insert()  依据数组修改，注意对应数组含义  3. query()  重点在改最内层的while循环 |
| 1. struct ACAuto { 2. const static int type = 26; 3. int next[MAXN][type], fail[MAXN], end[MAXN], len[MAXN]; 4. int root, tot; 5. int newnode() { 6. for( int i = 0; i < type; ++i ) next[tot][i] = -1; 7. len[tot] = 0; end[tot++] = -1; 8. return tot - 1; 9. } 10. void init() { 11. tot = 0; 12. root = newnode(); 13. } 14. void insert( char \*s ) { 15. int tlen = strlen( s ), u = root; 16. for( int i = 0; i < tlen; ++i ) { 17. int idx = s[i] - 'a'; 18. if( next[u][idx] == -1 ) next[u][idx] = newnode(); 19. u = next[u][idx]; 20. } 21. end[u] = 1; len[u] = tlen; 22. } 23. void build() { 24. queue<int> que; 25. fail[root] = root; 26. for( int i = 0; i < type; ++i ) { 27. if( next[root][i] == -1 ) next[root][i] = root; 28. else { 29. fail[next[root][i]] = root; 30. que.push( next[root][i] ); 31. } 32. } 33. while( !que.empty() ) { 34. int u = que.front(); que.pop(); 35. for( int i = 0; i < type; ++i ) { 36. if( next[u][i] == -1 ) next[u][i] = next[fail[u]][i]; 37. else { 38. fail[next[u][i]] = next[fail[u]][i]; 39. que.push( next[u][i] ); 40. } 41. } 42. } 43. } 44. void query( char \*s ) { 45. int idx, tlen = strlen( s ), u = root; 46. memset( pos, 0, sizeof pos ); 47. for( int i = 0; i < tlen; ++i ) { 48. // 忽略大小写 49. if( s[i] >= 'A' && s[i] <='Z' ) idx = s[i]-'A'; 50. else if( s[i] >= 'a' && s[i] <='z' ) idx = s[i]-'a'; 51. else continue; 52. u = next[u][idx]; 53. int tp = u; 54. while( tp != root ) { 55. if( end[tp] != -1 ) { 56. pos[i + 1] -= 1; 57. pos[i - len[tp] + 1] += 1; 58. break; 59. } 60. tp = fail[tp]; 61. } 62. } 63. int cnt = 0; 64. for( int i = 0; i < tlen; ++i ) { 65. cnt += pos[i]; 66. if( cnt <= 0 ) putchar( s[i] ); 67. else putchar( '\*' ); 68. } 69. puts( "" ); 70. } 71. }AC; |

## KMP

|  |
| --- |
| 1. #include <bits/stdc++.h> 2. using namespace std; 3. int main() { 4. int i, j; 5. int p[100010], len1, len2; 6. char A[100010], B[100010]; 7. scanf( "%s%s", A, B ); 8. len1 = strlen( A ); 9. len2 = strlen( B ); 10. p[0] = -1; 11. for( i = 1, j = -1; i < len2; ++i ) { 12. while( ( j >= 0 ) && ( B[j + 1] != B[i] ) ) j = p[j]; 13. if( B[j + 1] == B[i] ) ++j; 14. p[i] = j; 15. } 16. for( i = 0, j = -1; i < len1; ++i ) { 17. while( ( j >= 0 ) && ( B[j + 1] != A[i] ) ) j = p[j]; 18. if( B[j + 1] == A[i] ) ++j; 19. if( j == len2 - 1 ) { 20. cout << i + 1 - len2 << endl; 21. // j = p[j]; //multiple matching 22. break; 23. } 24. } 25. return 0; 26. } |

# 数学

## Ploya定理

|  |
| --- |
| 1. #define LL long long 2. LL c, s; 3. LL gcd( LL a, LL b ) { return b == 0 ? a : gcd( b, a % b ); } 4. LL pow( LL a, LL b ) { 5. LL ans = 1; 6. while( b ) { 7. if( b & 1 ) ans \*= a; 8. a \*= a; 9. b >>= 1; 10. } 11. return ans; 12. } 13. LL polya() { 14. LL i, j; 15. LL sum = 0; 16. for( i = 1; i <= s; i++ ) 17. sum += pow( c, gcd( s, i ) ); 18. if( s & 1 ) 19. sum += s \* pow( c, s / 2 + 1 ); 20. else 21. sum += ( ( s / 2 ) \* pow( c, s / 2 ) ) + ( ( s / 2 ) \* pow( c, s / 2 + 1 ) ); 22. sum /= ( 2 \* s ); 23. return sum; 24. } 25. int main() { 26. while( ~scanf( "%I64d%I64d", &c, &s ) && ( c || s ) ) 27. printf( "%I64d\n", polya() ); 28. return 0; 29. } |

## SG博弈

|  |
| --- |
| 定义P-position和N-position，其中P代表Previous，N代表Next。直观的说，上一次move的人有必胜策略的局面是P-position，也就是“后手可保证必胜”或者“先手必败”，现在轮到move的人有必胜策略的局面是N-position，也就是“先手可保证必胜”。更严谨的定义是：1.无法进行任何移动的局面（也就是terminal position）是P-position；2.可以移动到P-position的局面是N-position；3.所有移动都导致N-position的局面是P-position。 按照这个定义，如果局面不可能重现，或者说positions的集合可以进行拓扑排序，那么每个position或者是P-position或者是N-position，而且可以通过定义计算出来。  !!!对于一个Nim游戏的局面(a1,a2,...,an)，它是P-position当且仅当a1^a2^...^an=0，其中^表示异或(xor)运算。  如果Nim游戏中的规则稍微变动一下,每次最多只能取K个,怎么处理? 方法是将每堆石子数mod (k+1). |
| 1. const int MAXN = 1e5 + 10; 2. int sg[MAXN], a[MAXN]; 3. int n, m; 4. int gao( int x ) { 5. if( ~sg[x] ) return sg[x]; 6. int i, vis[100]; 7. memset( vis, 0, sizeof( vis ) ); 8. for( i = 0; i < n; ++i ) { 9. if( x < a[i] ) break; 10. vis[gao( x - a[i] )] = 1; 11. } 12. for( i = 0; vis[i]; ++i ); 13. return sg[x] = i; } 14. int main() { 15. int th, ta, ans; 16. while( ~scanf( "%d", &n ) && n ) { 17. memset( sg, -1, sizeof( sg ) ); 18. for( int i = 0; i < n; ++i ) scanf( "%d", a + i ); 19. sort( a, a + n ); 20. scanf( "%d", &m ); 21. while( m-- ) { 22. ans = 0; 23. scanf( "%d", &th ); 24. while( th-- ) { 25. scanf( "%d", &ta ); 26. ans ^= gao( ta ); 27. } 28. putchar( ans ? 'W' : 'L' ); 29. } 30. puts( "" ); 31. } 32. return 0; 33. } 34. /\* 35. const int MAXN = 1e6 + 10; 36. int n; 37. int main() { 38. int t, ta; 39. scanf( "%d", &t ); 40. while( t-- ) { 41. int ans = 0; 42. scanf( "%d", &n ); 43. for( int i = 0; i < n; ++i ) { 44. scanf( "%d", &ta ); 45. if( ta % 4 == 0 ) ans ^= ( ta - 1 ); 46. else if( ta % 4 == 1 || ta % 4 == 2 ) ans ^= ta; 47. else ans ^= ( ta + 1 ); 48. } 49. if( ans ) puts( "Alice" ); 50. else puts( "Bob" ); 51. } 52. return 0; 53. } 54. \*/ |

## 逆元处理求组合数(MOD是质数)

|  |
| --- |
| 1. typedef long long LL; 2. const int INF = 0x7F7F7F7F; 3. const int MOD = 1e9 + 7; 4. const int MAXN = 100000 + 10; 5. LL fac[MAXN], inv[MAXN]; 6. LL pmod( LL a, LL b ) { 7. LL ans = 1; 8. while( b ) { 9. if( b & 1 ) ans = ans \* a % MOD; 10. a = a \* a % MOD; 11. b >>= 1; 12. } 13. return ans; 14. } 15. LL \_inv( LL x ) { return pmod( x, MOD - 2 ); } 16. int main() { 17. fac[0] = inv[0] = 1; 18. for( int i = 1; i < MAXN; ++i ) { fac[i] = fac[i - 1] \* i; inv[i] = \_inv( fac[i] ); } 19. return 0; 20. } |

## 组合数

|  |
| --- |
| 1. const int M = 1007; 2. const int MAXN = 1000; 3. long long C[MAXN+1][MAXN+1]; 4. void Initial() { 5. int i,j; 6. for(i=0; i<=MAXN; ++i) { 7. C[0][i] = 0; 8. C[i][0] = 1; 9. } 10. for(i=1; i<=MAXN; ++i) { 11. for(j=1; j<=MAXN; ++j) 12. C[i][j] = (C[i-1][j] + C[i-1][j-1]); 13. } 14. } 15. long long Combination(int n, int m) { return C[n][m]; } 16. int main() { 17. int T,i,m,n; 18. Initial(); 19. while( ~scanf("%d%d",&n,&m) ){ 20. printf("C(%d，%d)=%I64d\n",n,m,Combination(n,m)); 21. } 22. return 0; 23. } |

## 快速线性素数筛

|  |
| --- |
| 1. #define N 1000010 2. int prime[N]; 3. bool vis[N]; 4. int num; 5. void \_prime() { 6. int i, j; 7. memset( vis, true, sizeof( vis ) ); 8. vis[0] = vis[1] = 0; 9. for( i = 2, num = 0; i < N; ++i ) { 10. if( vis[i] ) 11. prime[num++] = i; 12. for( j = 0; j < num && i \* prime[j] < N; ++j ) { 13. vis[i \* prime[j]] = 0; 14. if( !( i % prime[j] ) ) 15. break; 16. } 17. } 18. return ; 19. } |

# 计算几何

## 点与矩阵最小距离

|  |
| --- |
| 1. const double eps = 1e-8; 2. const double PI = acos( -1.0 ); 3. int sig( double x ) { 4. if( fabs( x ) < eps ) return 0; 5. return x > 0 ? 1 : -1; 6. } 7. struct Point { 8. double x, y; 9. Point() {} 10. Point( const double xx, const double yy ) { x = xx; y = yy; } 11. Point operator + ( const Point &tp ) const { return Point( x + tp.x, y + tp.y ); } 12. Point operator - ( const Point &tp ) const { return Point( x - tp.x, y - tp.y ); } 13. double operator \* ( const Point &tp ) const { return x \* tp.x + y \* tp.y; } 14. double operator ^ ( const Point &tp ) const { return x \* tp.y - y \* tp.x; } 15. bool operator < ( const Point &tp ) const { 16. if( sig( x - tp.x ) ) return sig( x - tp.x ) < 0; 17. else return sig( y - tp.y ) < 0; 18. } 19. }; 20. Point poi[5], cir, src; 21. double r; 22. double dist( const Point &a, const Point &b ) { 23. double dx = a.x - b.x; 24. double dy = a.y - b.y; 25. return sqrt( dx \* dx + dy \* dy ); 26. } 27. double dptol( const Point &a, const Point &b, const Point &c ) { 28. double ret = 0; 29. if( sig( ( c - b ) \* ( a - b ) ) > 0 && sig( ( b - c ) \* ( a - c ) ) > 0 ) 30. ret = fabs( ( b - a ) ^ ( c - a ) ) / dist( b, c ); 31. else ret = min( dist( a, b ), dist( a, c ) ); 32. return ret; 33. } 34. double dptor( const Point &a ) { 35. double d1 = min( dptol( a, poi[0], poi[1] ), dptol( a, poi[0], poi[2] ) ); 36. double d2 = min( dptol( a, poi[1], poi[3] ), dptol( a, poi[2], poi[3] ) ); 37. return min( d1, d2 ); 38. } 39. int main() { 40. return 0; 41. } |

## 点在多边形内

|  |
| --- |
| 1. const double eps = 1e-8; 2. const double PI = acos( -1.0 ); 3. int sig( double x ) { 4. if( fabs( x ) < eps ) return 0; 5. return x > 0 ? 1 : -1; 6. } 7. struct Point { 8. double x, y; 9. Point() {} 10. Point( const double xx, const double yy ) { x = xx; y = yy; } 11. Point operator + ( const Point &tp ) const { return Point( x + tp.x, y + tp.y ); } 12. Point operator - ( const Point &tp ) const { return Point( x - tp.x, y - tp.y ); } 13. double operator \* ( const Point &tp ) const { return x \* tp.x + y \* tp.y; } 14. double operator ^ ( const Point &tp ) const { return x \* tp.y - y \* tp.x; } 15. bool operator < ( const Point &tp ) const { 16. if( sig( x - tp.x ) ) return sig( x - tp.x ) < 0; 17. else return sig( y - tp.y ) < 0; 18. } 19. }; 20. typedef Point pVector; 21. typedef vector<Point> Polygon; 22. bool OnSegment( Point p, Point a, Point b ) { 23. if( sig( ( p - a ) \* ( p - b ) ) ) return 0; 24. return sig( a.x - p.x ) \* sig( b.x - p.x ) <= 0 && sig( a.y - p.y ) \* sig( b.y - p.y ) <= 0; 25. } 26. int isPointInPolygon( Point p, Polygon poly ) { 27. int wn = 0; 28. int n = poly.size(); 29. for( int i = 0; i < n; ++i ) { 30. if( OnSegment( p, poly[i], poly[( i + 1 ) % n] ) ) return 0; 31. int k = sig( ( poly[( i + 1 ) % n] - poly[i] ) ^ ( p - poly[i] ) ); 32. int d1 = sig( poly[i].y - p.y ); 33. int d2 = sig( poly[( i + 1 ) % n].y - p.y ); 34. if( k > 0 && d1 <= 0 && d2 > 0 ) ++wn; 35. if( k < 0 && d2 <= 0 && d1 > 0 ) --wn; 36. } 37. return wn; 38. } 39. int main() { 40. return 0; 41. } |

## 多边形与圆面积交

|  |
| --- |
| 1. typedef long long LL; 2. const double INF = 1000000000000; 3. const double eps = 1e-12; 4. const double PI = acos( -1.0 ); 5. const int MAXN = 100009; 6. const int MOD = 1000000007; 7. struct Point { 8. double x,y; 9. Point(){} 10. Point( double xx, double yy ) { x = xx; y = yy; } 11. Point operator - ( Point s ) { return Point( x - s.x, y - s.y ); } 12. Point operator + ( Point s ) { return Point( x + s.x, y + s.y ); } 13. double operator \* ( Point s ) { return x \* s.x + y \* s.y; } 14. double operator ^ ( Point s ) { return x \* s.y - y \* s.x; } 15. }poi[MAXN]; 16. double mmax( double a, double b ) { return a > b ? a : b; } 17. double mmin( double a, double b ) { return a < b ? a : b; } 18. double len( Point a ) { return sqrt( a \* a ); } 19. double dist( Point a, Point b ) { return len( b - a ); } 20. double cross( Point a, Point b, Point c ) { return ( b - a ) ^ ( c - a ); } 21. double dot( Point a, Point b, Point c ) { return ( b - a ) \* ( c - a ); } 22. double area( Point b, Point c, double r ) { 23. Point a( 0.0, 0.0 ); 24. if( dist( b, c ) < eps ) return 0.0; 25. double h = fabs( cross( a, b, c ) ) / dist( b, c ); 26. //两个端点都在圆的外面则分为两种情况 27. //两个端点都在圆内的情况 28. //一个端点在圆上一个端点在圆内的情况 29. if( dist( a, b ) > r - eps && dist( a, c ) > r - eps ) { 30. double angle = acos( dot( a, b, c ) / dist( a, b ) / dist( a, c ) ); 31. if( h > r - eps ) return 0.5 \* r \* r \* angle; 32. else if( dot( b, a, c ) > 0 && dot( c, a, b ) > 0 ) { 33. double angle1 = 2 \* acos( h / r ); 34. return 0.5 \* r \* r \* fabs( angle - angle1 ) + 0.5 \* r \* r \* sin( angle1 ); 35. } else return 0.5 \* r \* r \* angle; 36. } else if( dist( a, b ) < r + eps && dist( a, c) < r + eps ) { 37. return 0.5 \* fabs( cross( a, b, c ) ); 38. } else { 39. //默认b在圆内 40. if( dist( a, b ) > dist( a, c ) ) swap(b,c); 41. //ab距离为0直接返回0 42. if( fabs( dist( a, b ) ) < eps ) return 0.0; 43. if( dot( b, a, c ) < eps ) { 44. double angle1 = acos( h / dist( a, b ) ); 45. double angle2 = acos( h / r ) - angle1; 46. double angle3 = acos( h / dist( a, c ) ) - acos( h / r ); 47. return 0.5 \* dist( a, b ) \* r \* sin( angle2 ) + 0.5 \* r \* r \* angle3; 48. } else { 49. double angle1 = acos( h / dist( a, b ) ); 50. double angle2 = acos( h / r ); 51. double angle3 = acos( h / dist( a, c ) ) - angle2; 52. return 0.5 \* r \* dist( a, b ) \* sin( angle1 + angle2 ) + 0.5 \* r \* r \* angle3; 53. } 54. } 55. return 0.0; 56. } 57. int main() { 58. int n; 59. double x, y, v, ang, t, g, r; 60. while( ~scanf( "%lf%lf%lf%lf%lf%lf%lf", &x, &y, &v, &ang, &t, &g, &r ) && 61. x + y + v + ang + t + g + r ) { 62. scanf( "%d", &n ); 63. for( int i = 0; i < n; ++i ) { 64. scanf( "%lf%lf", &poi[i].x, &poi[i].y ); 65. } 66. poi[n] = poi[0]; 67. Point O( x, y ); 68. double tmp = sin( ang / 180 \* PI ); 69. O.x += v \* t \* cos( ang / 180 \* PI ); 70. if( t \* g <= v ) O.y += ( v \* tmp + ( v\*tmp - g\*t ) ) / 2\*t; 71. else { 72. double tt = v \* tmp / g; 73. O.y += ( v \* tmp / 2 ) \* tt; 74. tt = t - tt; 75. O.y -= ( g \* tt \* tt ) / 2; 76. } 77. for( int i = 0; i <= n; ++i ) poi[i] = poi[i] - O; 78. O = Point( 0, 0 ); 79. double sum = 0; 80. for( int i = 0; i < n; ++i ) { 81. int j = i + 1; 82. double s = area( poi[i], poi[j], r ); 83. if( cross( O, poi[i], poi[j] ) > 0 ) sum += s; 84. else sum -= s; 85. } 86. printf( "%.2lf\n", fabs( sum ) ); 87. } 88. return 0; 89. } |

## 矩形面积并

|  |
| --- |
| 1. const double eps = 1e-10; 2. int n; 3. pair<double, int> c[10000]; 4. struct point { double x, y; } p[600][5]; 5. int dblcmake\_pair( double x ) { 6. if( fabs(x) < eps ) return 0; 7. return x > 0 ? 1 : -1; 8. } 9. double cross( point& p1, point& p2, point& p3 ) { 10. return (p2.x-p1.x)\*(p3.y-p1.y) - (p2.y-p1.y)\*(p3.x-p1.x); 11. } 12. double dot( point aa, point bb ) { 13. return aa.x\*bb.x + aa.y\*bb.y; 14. } 15. double segP( point p1, point p2, point p3 ) { 16. if( dblcmake\_pair(p2.x-p3.x) ) 17. return (p1.x-p2.x)/(p3.x-p2.x); 18. else 19. return (p1.y-p2.y)/(p3.y-p2.y); 20. } 21. double polyUnion() { 22. int i, j, ii, jj, ta, tb, r, d; 23. double z, w, s, sum, tc, td; 24. point tmake\_pair1, tmake\_pair2; 25. sum = 0; 26. for( i = 0; i < n; ++i ) { 27. for( ii = 0; ii < 4; ++ii ) { 28. r = 0; 29. c[r++] = make\_pair(0., 0); 30. c[r++] = make\_pair(1., 0); 31. for( j = 0; j < n; ++j ) if( i-j ) { 32. for( jj = 0; jj < 4; ++jj ) { 33. ta = dblcmake\_pair( cross(p[i][ii], p[i][ii+1], p[j][jj]) ); 34. tb = dblcmake\_pair( cross(p[i][ii], p[i][ii+1], p[j][jj+1]) ); 35. if( !ta && !tb ) { 36. tmake\_pair1.x = p[j][jj+1].x-p[j][jj].x; 37. tmake\_pair1.y = p[j][jj+1].y-p[j][jj].y; 38. tmake\_pair2.x = p[i][ii+1].x-p[i][ii].x; 39. tmake\_pair2.y = p[i][ii+1].y-p[i][ii].y; 40. if( dblcmake\_pair( dot(tmake\_pair1, tmake\_pair2) ) > 0 && j < i ) { 41. c[r++] = make\_pair( segP(p[j][jj], p[i][ii], p[i][ii+1]), 1 ); 42. c[r++] = make\_pair( segP(p[j][jj+1], p[i][ii], p[i][ii+1]), -1 ); 43. } 44. } else if( ta >= 0 && tb < 0 ) { 45. tc = cross(p[j][jj], p[j][jj+1], p[i][ii]); 46. td = cross(p[j][jj], p[j][jj+1], p[i][ii+1]); 47. c[r++] = make\_pair(tc/(tc-td), 1); 48. } else if( ta < 0 && tb >= 0 ) { 49. tc = cross(p[j][jj], p[j][jj+1], p[i][ii]); 50. td = cross(p[j][jj], p[j][jj+1], p[i][ii+1]); 51. c[r++] = make\_pair(tc/(tc-td), -1); 52. } 53. } 54. } 55. sort(c, c+r); 56. z = min(max(c[0].first, 0.), 1.); 57. d = c[0].second; s = 0; 58. for( j = 1; j < r; ++j ) { 59. w = min(max(c[j].first, 0.), 1.); 60. if( !d ) s += w-z; 61. d += c[j].second; 62. z = w; 63. } 64. tmake\_pair1.x = tmake\_pair1.y = 0; 65. sum += cross(tmake\_pair1, p[i][ii], p[i][ii+1])\*s; 66. } 67. } 68. return 0.5\*sum; 69. } 70. int main() { 71. int i, j; 72. double area, tmake\_pair; 73. while( scanf("%d", &n) != EOF ) { 74. area = 0; 75. for( i = 0; i < n; ++i ) { 76. for( j = 0; j < 4; ++j ) 77. scanf("%lf %lf", &p[i][j].x, &p[i][j].y); 78. p[i][4] = p[i][0]; 79. tmake\_pair = 0; 80. for( j = 1; j <= 4; ++j ) 81. tmake\_pair += p[i][j-1].x\*p[i][j].y - p[i][j-1].y\*p[i][j].x; 82. area += fabs(tmake\_pair); 83. if( dblcmake\_pair(tmake\_pair) < 0 ) swap(p[i][1], p[i][3]); 84. } 85. printf("%.10lf\n", 0.5\*area/polyUnion() ); 86. } 87. return 0; 88. } |

## 凸多边形面积并

|  |
| --- |
| 1. const int maxn = 505; 2. const int maxm = 5005; 3. const double eps = 1e-8; 4. const double PI = acos(-1.0); 5. inline int dcmp(double a) { 6. return a < -eps ? -1 : a > eps; 7. } 8. struct Point { 9. double x, y; 10. Point(){} 11. Point(double a, double b): x(a), y(b){} 12. bool operator < (const Point p) const { 13. return y+eps < p.y || (y < p.y+eps && x+eps < p.x); 14. } 15. bool operator == (const Point p) const { 16. return !dcmp(x-p.x) && !dcmp(y-p.y); 17. } 18. Point operator + (const Point p) const { 19. return Point(x+p.x, y+p.y); 20. } 21. Point operator - (const Point p) const { 22. return Point(x-p.x, y-p.y); 23. } 24. Point operator \* (const double p) const { 25. return Point(x\*p, y\*p); 26. } 27. Point operator / (const double p) const { 28. return Point(x/p, y/p); 29. } 30. double operator \* (const Point p) const { 31. return x\*p.y - y\*p.x; 32. } 33. double operator / (const Point p) const { 34. return x\*p.x + y\*p.y; 35. } 36. void input() { scanf("%lf %lf", &x, &y); } 37. }; 38. struct Polygon { 39. int n; 40. Point p[maxn]; 41. Point& operator [] (const int i) { return p[i]; } 42. void init() { 43. double x1, x2, y1, y2; 44. n = 4; 45. for(int i = 0; i < 4; i++) p[i].input(); 46. } 47. double Area() { 48. double sum = 0; 49. for(int i = 1; i < n-1; i++) 50. sum += (p[i]-p[0]) \* (p[i+1]-p[0]); 51. return sum / 2.0; 52. } 53. }; 54. struct Polygons { 55. int n; 56. Polygon py[maxn]; 57. std::pair <double, int> c[maxm]; 58. void init() { n = 0; } 59. void push(Polygon p) { 60. p[p.n] = p[0]; 61. py[n++] = p; 62. } 63. double seg(Point p, Point p1, Point p2) { 64. if(!dcmp(p1.x-p2.x)) 65. return (p.y-p1.y) / (p2.y-p1.y); 66. return (p.x-p1.x) / (p2.x-p1.x); 67. } 68. double PolyUnion() { 69. int d, r, ta, tb; 70. double s, w, z, sum, tc, td; 71. sum = 0; 72. for(int i = 0; i < n; i++) { 73. for(int ii = 0; ii < py[i].n; ii++) { 74. r = 0; 75. c[r++] = make\_pair(0.0, 0); 76. c[r++] = make\_pair(1.0, 0); 77. for(int j = 0; j < n; j++) { 78. if(i == j) continue; 79. for(int jj = 0; jj < py[j].n; jj++) { 80. ta = dcmp((py[i][ii+1]-py[i][ii])\*(py[j][jj]-py[i][ii])); 81. tb = dcmp((py[i][ii+1]-py[i][ii])\*(py[j][jj+1]-py[i][ii])); 82. if(!ta && !tb) { 83. if((py[j][jj+1]-py[j][jj])/(py[i][ii+1]-py[i][ii]) > 0 && j < i) { 84. c[r++] = make\_pair(seg(py[j][jj], py[i][ii], py[i][ii+1]), 1); 85. c[r++] = make\_pair(seg(py[j][jj+1], py[i][ii], py[i][ii+1]), -1); 86. } 87. } else if(ta >= 0 && tb < 0) { 88. tc = (py[j][jj+1]-py[j][jj]) \* (py[i][ii]-py[j][jj]); 89. td = (py[j][jj+1]-py[j][jj]) \* (py[i][ii+1]-py[j][jj]); 90. c[r++] = make\_pair(tc/(tc-td), 1); 91. } else if(ta < 0 && tb >= 0) { 92. tc = (py[j][jj+1]-py[j][jj]) \* (py[i][ii]-py[j][jj]); 93. td = (py[j][jj+1]-py[j][jj]) \* (py[i][ii+1]-py[j][jj]); 94. c[r++] = make\_pair(tc/(tc-td), -1); 95. } 96. } 97. } 98. std::sort(c, c+r); 99. z = std::min(std::max(c[0].first, 0.0), 1.0); 100. d = c[0].second; 101. s = 0; 102. for(int j = 1; j < r; j++) { 103. w = std::min(std::max(c[j].first, 0.0), 1.0); 104. if(!d) s += w - z; 105. d += c[j].second; 106. z = w; 107. } 108. sum += (py[i][ii]\*py[i][ii+1]) \* s; 109. } 110. } 111. return sum / 2.0; 112. } 113. }; 114. Polygons P; 115. Polygon pp; 116. int main() { 117. int n; 118. double area, sum = 0; 119. scanf("%d",&n); 120. P.init(); 121. for(int i = 0; i < n; i++) { 122. pp.init(); 123. area = pp.Area(); 124. if(area < 0) { 125. for(int j = 0,k = pp.n-1; j < k; j++, k--) 126. std::swap(pp[j], pp[k]); 127. area = -area; 128. } 129. sum += area; 130. P.push(pp); 131. } 132. printf("%.10f\n", sum / P.PolyUnion()); 133. return 0; 134. } |

## 旋转卡壳

|  |
| --- |
| //计算凸包直径，输入凸包ch，顶点个数为n，按逆时针排列，输出直径的平方 |
| 1. int rotating\_calipers(Point \*ch,int n){ 2. int q=1,ans=0; 3. ch[n]=ch[0]; 4. for(int p=0;p<n;p++) { 5. while(cross(ch[p+1],ch[q+1],ch[p])>cross(ch[p+1],ch[q],ch[p])) 6. q=(q+1)%n; 7. ans=max(ans,max(dist2(ch[p],ch[q]),dist2(ch[p+1],ch[q+1]))); 8. } 9. return ans; 10. } |

## 圆的面积并(辛普森积分法)

|  |
| --- |
| 1. int n,top,st,ed; 2. ld xl[1001],xr[1001]; 3. ld ans; 4. bool del[1001]; 5. struct data{ld x,y,r;}t[1001],sk[1001]; 6. struct line{ld l,r;}p[1001]; 7. ld dis(data a,data b) 8. {return sqrt((a.x-b.x)\*(a.x-b.x)+(a.y-b.y)\*(a.y-b.y));} 9. bool cmp1(data a,data b){return a.r<b.r;} 10. bool cmp2(data a,data b){return a.x-a.r<b.x-b.r;} 11. bool cmp3(line a,line b){return a.l<b.l;} 12. void ini() { 13. scanf("%d",&n); 14. for(int i=1;i<=n;i++) 15. {scanf("%lf%lf%lf",&t[i].x,&t[i].y,&t[i].r);} 16. sort(t+1,t+n+1,cmp1); 17. for(int i=1;i<=n;i++) 18. for(int j=i+1;j<=n;j++) 19. if(dis(t[i],t[j])<=t[j].r-t[i].r) 20. {del[i]=1;break;} 21. for(int i=1;i<=n;i++)if(!del[i])sk[++top]=t[i];n=top; 22. sort(sk+1,sk+n+1,cmp2); 23. } 24. ld getf(ld x) { 25. int sz=0,i,j;ld r,len=0,dis; 26. for(i=st;i<=ed;i++) { 27. if(x<=xl[i]||x>=xr[i])continue; 28. dis=sqrt(sk[i].r-(x-sk[i].x)\*(x-sk[i].x)); 29. p[++sz].l=sk[i].y-dis;p[sz].r=sk[i].y+dis; 30. } 31. sort(p+1,p+sz+1,cmp3); 32. for(i=1;i<=sz;i++) { 33. r=p[i].r; 34. for(j=i+1;j<=sz;j++) { 35. if(p[j].l>r)break; 36. if(r<p[j].r)r=p[j].r; 37. } 38. len+=r-p[i].l;i=j-1; 39. } 40. return len; 41. } 42. ld cal(ld l,ld fl,ld fmid,ld fr) 43. {return (fl+fmid\*4+fr)\*l/6;} 44. ld simpson(ld l,ld mid,ld r,ld fl,ld fmid,ld fr,ld s) 45. { 46. ld m1=(l+mid)/2,m2=(r+mid)/2; 47. ld f1=getf(m1),f2=getf(m2); 48. ld g1=cal(mid-l,fl,f1,fmid),g2=cal(r-mid,fmid,f2,fr); 49. if(fabs(g1+g2-s)<eps)return g1+g2; 50. return simpson(l,m1,mid,fl,f1,fmid,g1)+simpson(mid,m2,r,fmid,f2,fr,g2); 51. } 52. void work() { 53. int i,j;ld l,r,mid,fl,fr,fmid; 54. for(i=1;i<=n;i++){ 55. l[i]=sk[i].x-sk[i].r; 56. r[i]=sk[i].x+sk[i].r; 57. k[i].r\*=sk[i].r; 58. } 59. for(i=1;i<=n;i++) { 60. l=xl[i];r=xr[i]; 61. for(j=i+1;j<=n;j++) { 62. if(xl[j]>r)break; 63. if(xr[j]>r)r=xr[j]; 64. } 65. st=i;ed=j-1;i=j-1; 66. mid=(l+r)/2; 67. fl=getf(l);fr=getf(r);fmid=getf(mid); 68. ans+=simpson(l,mid,r,fl,fmid,fr,cal(r-l,fl,fmid,fr)); 69. } 70. } 71. int main() { 72. ini(); 73. work(); 74. printf("%.3lf",ans); 75. return 0; 76. } |

# 杂七杂八

## LIS

|  |
| --- |
| 1. int n, a[20010]; 2. int c[20010]; 3. int len = 0; 4. int find( int x ) { 5. int l = 1, r = len, mid; 6. while( l <= r ) { 7. mid = ( l + r ) >> 1; 8. if( x > c[mid] ) l = mid + 1; //求上升序列，就表示x更大，那么就是大于 9. else r = mid - 1; 10. } 11. return l; 12. } 13. int main() { 14. scanf( "%d", &n ); 15. for( int i = 1; i <= n; i++ ) 16. scanf( "%d", &a[i] ); 17. for( int i = 1; i <= n; i++ ) { 18. int k = find( a[i] ); 19. c[k] = a[i]; 20. len = max( len, k ); 21. } 22. printf( "%d", len ); 23. return 0; 24. } |

## 斯坦纳树

|  |
| --- |
| 1. #define INF 0x3F3F3F3F 2. const int sta[]={ 0, 3, 12, 48, 192, 15, 51, 195, 60, 204, 240, 63, 207, 243, 252, 255 }; 3. int dp[300][35], dis[35][35], info[10]; 4. map<string, int> nmap; 5. int lowbit( int x ) { 6. return ( x & ( -x ) ); 7. } 8. int bit( int x ) { 9. x = lowbit( x ); 10. int res; 11. for( res = 0; x; x >>= 1, ++res ); 12. return res - 1; 13. } 14. int main() { 15. int n, m, ans; 16. int ta, tb, c; 17. string s, t; 18. while( ~scanf( "%d%d", &n, &m ) && ( n || m ) ) { 19. memset( dis, 0x3F, sizeof( dis ) ); 20. memset( dp, 0x3F, sizeof( dp ) ); 21. nmap.clear(); 22. for( int i = 0; i < n; ++i ) { 23. cin >> s; 24. nmap[s] = i; 25. dis[i][i] = 0; 26. } 27. for( int i = 0; i < m; ++i ) { 28. cin >> s >> t >> c; 29. ta = nmap[s]; tb = nmap[t]; 30. dis[ta][tb] = dis[tb][ta] = min( dis[ta][tb], c ); 31. } 32. for( int k = 0; k < n; ++k ) { 33. for( int i = 0; i < n; ++i ) 34. for( int j = 0; j < n; ++j ) 35. dis[i][j] = min( dis[i][j], dis[i][k] + dis[k][j] ); 36. } 37. for( int i = 0; i < 8; ++i ) { 38. cin >> s; 39. info[i] = nmap[s]; 40. for( int j = 0; j < n; ++j ) { 41. dp[1 << i][j] = dis[info[i]][j]; 42. } 43. } 44. for( int i = 0; i < 256; ++i ) { 45. if( i & ( i - 1 ) == 0 ) continue; 46. c = 0; 47. for( int j = 0; j < n; ++j ) { 48. for( int sub = i; sub; sub = ( sub - 1 ) & i ) { 49. dp[i][j] = min( dp[i][j], dp[sub][j] + dp[i - sub][j] ); 50. } 51. if( dp[i][j] < dp[i][c] ) c = j; 52. } 53. for( int j = 0; j < n; ++j ) { 54. for( int k = 0; k < n; ++k ) { 55. dp[i][k] = min( dp[i][k], dp[i][j] + dis[j][k] ); 56. } 57. } 58. } 59. ans = INF; 60. for( int p1 = 0; p1 < 16; ++p1 ) { 61. for( int p2 = 0; p2 < 16; ++p2 ) { 62. for( int p3 = 0; p3 < 16; ++p3 ) { 63. for ( int p4 = 0; p4 < 16; ++p4 ) { 64. if( sta[p1] + sta[p2] + sta[p3] + sta[p4] == 255 ) { 65. for( int i = 0; i < n; ++i ) { 66. int tmp = 0; 67. if( sta[p1] ) tmp += dp[sta[p1]][info[bit( sta[p1] )]]; 68. if( sta[p2] ) tmp += dp[sta[p2]][info[bit( sta[p2] )]]; 69. if( sta[p3] ) tmp += dp[sta[p3]][info[bit( sta[p3] )]]; 70. if( sta[p4] ) tmp += dp[sta[p4]][info[bit( sta[p4] )]]; 71. ans = min( ans, tmp ); 72. } 73. } 74. } 75. } 76. } 77. } 78. printf( "%d\n", ans ); 79. } 80. return 0; 81. } |

## 归并排序求逆序对

|  |
| --- |
| 1. #include <iostream> 2. #include <cstdio> 3. using namespace std; 4. #define MAXN 500010 5. #define INF 0x3FFFFFFF 6. int L[MAXN], R[MAXN], a[MAXN]; 7. long long cnt; 8. void \_merge( int l, int m, int r ) { 9. int i, j, k; 10. int n1, n2; 11. n1 = m - l + 1; 12. n2 = r - m; 13. for( i = 0; i < n1; ++i ) 14. L[i] = a[l + i]; 15. for( i = 0; i < n2; ++i ) 16. R[i] = a[m + 1 + i]; 17. L[n1] = INF; 18. R[n2] = INF; 19. i = j = 0; 20. for( k = l; k <= r; ++k ) { 21. if( L[i] <= R[j] ) 22. a[k] = L[i++]; 23. else { 24. a[k] = R[j++]; 25. cnt += n1 - i; 26. } 27. } 28. return ; 29. } 30. void \_merge\_sort( int l, int r ) { 31. if( l < r ) { 32. int m = ( l + r ) / 2; 33. \_merge\_sort( l, m ); 34. \_merge\_sort( m + 1, r ); 35. \_merge( l, m, r ); 36. } 37. return ; 38. } 39. int main() { 40. int i, n; 41. while( ~scanf( "%d", &n ) && n ) { 42. cnt = 0; 43. for( i = 0; i < n; ++i ) 44. scanf( "%d", a + i ); 45. \_merge\_sort( 0, n - 1 ); 46. printf( "%lld\n", cnt ); 47. } 48. return 0; 49. } |

# 环境

## \_vimrc配置文件

|  |
| --- |
| syntax on  set ai  set softtabstop=4  set shiftwidth=4  set tabstop=4  set expandtab  set hls  set nocompatible  set wrap  set cindent  set number  set nobackup  set backspace=eol,indent,start  set guifont=Droid\ Sans\ Mono\ 12  nnoremap <C-h> <C-w>h  nnoremap <C-j> <C-w>j  nnoremap <C-k> <C-w>k  nnoremap <C-l> <C-w>l  nmap wx <C-w>x  nmap <F7> :vsp %<.in <CR>  nmap <F12> :!python <CR>  autocmd FileType c,cpp nmap <F8> <ESC>:w <CR><ESC>:!g++ % -std=c++11 -o2 -o %< <CR>  autocmd FileType c,cpp nmap <F9> :!./%< <%<.in <CR>  autocmd FileType c,cpp nmap <F10> :!./%< <CR>  autocmd FileType java nmap <F8> <ESC>:w <CR><ESC>:!javac %<.java <CR>  autocmd FileType java nmap <F9> :!java %< <%<.in <CR>  autocmd FileType java nmap <F10> :!java %< <CR>  "设置配色方案  colorscheme desert |

## CB配置

|  |
| --- |
| terminal： gnome-terminal -t $TITLE -x 在codeblocks-->setting-->环境变量 |