

ArosTemplate

(2013.03.14 updated)

目录	
InSkill	2
Ubuntu 下 CodeBlocks 更改调试终端	2
HDU 上的 DFS 爆栈问题的简易解决方法	2
通过内嵌汇编把堆空间作为栈空间使用_hdu_4118	2
Graph	2
spfa	2
二维最短路_hdu_4396	3
找环_hdu_4337	4
最小生成树的最佳替换边_hdu_4126	4
最小树形图_hdu_4009	5
Network	7
最大流 ISAP_hdu_3879	7
最大流-邻接表	8
最大流-邻接矩阵	8
最小费用最大流-邻接表	9
最小费用最大流-邻接矩阵	9
Number	10
组合数 C(N, R)	10
Structure	10
AC 自动机	10
AC 自动机_hdu_2222	10
AC 自动机+DP_hdu_2825	11
AC 自动机+概率 DP_hdu_3689	13
AC 自动机+矩阵_poj_2778	14
DP	15
离散 DP_hdu_4028	15
区间 DP_hdu_4293_1	16
树形背包 DP_hdu_4276	16
KMP	17
扩展 KMP_hdu_4300	17
扩展 KMP_hdu_4333	18
大数	19
bign-bint	19
bign-lrj	21
bign-str	22
后缀数组	24
第 K 个子串_hdu_3553	24
多串子串并集_后缀数组_hdu_4416	26
最长重复不重叠子串_后缀数组+按 height 分组+二分_poj_1743	27
线段树	28
矩形并面积_离散化+扫描线+线段树_hdu_4419	28
线段树求矩形并周长_hdu_1828	29
线段树求体积并_hdu_3642	30
线段树区间修改单点查询_220B	31
最长上升子序列	32
二维 LIS+方案输出_sgu_521	32
某矩形的 LIS_bupt_394	33
最长上升子序列_poj_3903	33
Mahjong_hdu_4431	33
RMQ-ST	35
Trie 树_编辑距离阈值匹配_UVALive_4769	35
编辑距离+BK 树_hdu_4323	36
后缀自动机_SPOJ_LCS2	37
斯坦纳树_hdu_4085	38
最大非空连续和+方案_hdu_1003	39

InSkill

Ubuntu 下 CodeBlocks 更改调试终端

在环境设置里进行如下设置：把 Terminal to launch console programs 那个选项改成“gnome-terminal -t \$TITLE -x”，原来是“xterm -T \$TITLE -e”。

HDU 上的 DFS 爆栈问题的简易解决方法

在文件 gui 头处加上这么一句“#pragma comment(linker, "/STACK:1024000000,1024000000")”后面两个数字随便写，你觉得能过就好，另外不要超了栈内存的真正上限。基于 VC++ 的编译预处理命令，这个代码必须拿 C++ 来提交，所以 C++ 会出现的那种 long long 和 __int64 的问题也要注意。

通过内嵌汇编把堆空间作为栈空间使用_hdu_4118

```
01 #include<cstdio>
02 #include<cstring>
03 #include<algorithm>
04 using namespace std;
05 const int MAXN = 100000+5, MAXM = 200000+5;
06 int T, N, X, Y, Z;
07 int e, head[MAXN], next[MAXM], v[MAXM];
08 int cnt[MAXN];
09 long long w[MAXM], ans;
10 void addedge(int x, int y, int z)
11 {
12     v[e] = y; w[e] = z;
13     next[e] = head[x]; head[x] = e++;
14 }
15 void dfs(int u, int fa = 0)
16 {
17     cnt[u] = 1;
18     for (int i = head[u]; i != -1; i = next[i]) if (v[i] != fa)
19     {
20         dfs(v[i], u);
21         ans += min(cnt[v[i]], N-cnt[v[i]])*2*w[i];
22         cnt[u] += cnt[v[i]];
23     }
24 }
25
26 void call_dfs()
27 {
28     const int STACK_SIZE = 1<<23;
29     static char stack[STACK_SIZE];
30     int bak;
```

```
31     __asm__ __volatile__
32     (
33         "movl %%esp, %0\n"
34         "movl %1, %%esp\n":
35         "=g"(bak):
36         "g"(stack+STACK_SIZE-1):
37     );
38
39     dfs(1);
40
41     __asm__ __volatile__
42     (
43         "movl %0, %%esp\n":
44         :
45         "g"(bak):
46     );
47 }
48
49 int main()
50 {
51     scanf("%d", &T);
52     for (int cas = 1; cas <= T; cas++)
53     {
54         e = 0;
55         memset(head, -1, sizeof(head));
56         scanf("%d", &N);
57         for (int i = 1; i < N; i++)
58         {
59             scanf("%d%d%d", &X, &Y, &Z);
60             addedge(X, Y, Z);
61             addedge(Y, X, Z);
62         }
63         ans = 0;
64         call_dfs();
65         printf("Case #d: %I64d\n", cas, ans);
66     }
67     return 0;
68 }
```

Graph

spfa

```
01 #include<cstdio>
02 #include<cstring>
03 #include<queue>
04 using namespace std;
05 const int MAXN = 1000+5, MAXM = 1000+5;
06 const int INF = 0x3f3f3f3f;
```

```

07 int n, m, e, s;
08 int v[MAXM], next[MAXM], head[MAXN];
09 int w[MAXM], d[MAXN];
10 int inq_cnt[MAXN]; //存在负权回路时需要
11 bool inq[MAXN];
12 queue<int> Q;
13 void addedge(int x, int y, int z)
14 {
15     v[e] = y; w[e] = z;
16     next[e] = head[x]; head[x] = e;
17     e++;
18 }
19 bool spfa()
20 {
21     for (int i = 1; i <= n; i++)
22         d[i] = (i == s ? 0 : INF);
23     memset(inq, 0, sizeof(inq));
24     memset(inq_cnt, 0, sizeof(inq_cnt));
25     while (!Q.empty()) Q.pop();
26     Q.push(s);
27     inq[s] = 1;
28     inq_cnt[s]++;
29     while (!Q.empty())
30     {
31         int u = Q.front(); Q.pop();
32         inq[u] = 0;
33         for(int e = head[u]; e != -1; e = next[e])
34             if(d[v[e]] > d[u]+w[e])
35             {
36                 d[v[e]] = d[u]+w[e];
37                 if(!inq[v[e]])
38                 {
39                     Q.push(v[e]);
40                     inq[v[e]] = 1;
41                     inq_cnt[v[e]]++;
42                     if (inq_cnt[v[e]] > n)
43                         return 0;
44                 }
45             }
46     }
47     return 1;
48 }
49 int main()
50 {
51     freopen("input.txt", "r", stdin);
52     freopen("output.txt", "w", stdout);
53     memset(head, -1, sizeof(head));
54     e = 0;
55
56     return 0;
57 }

```

二维最短路_hdu_4396

```

01 /*
02 题意：求至少经过 K 条边，到达终点的最短路 (K<=50)。
03 思路：因为 K<=500，所以每个节点最多扩展成 50 个节点，最后一个节点表示到达该节点时经过
    的边数（收集到的木材/10）已经满足 K 值对应的要求。然后 spfa，每个节点表示为 (编号, 经过的
    边数)。
04 */
05 #include<cstdio>
06 #include<cstring>
07 #include<algorithm>
08 #include<queue>
09 using namespace std;
10 const int MAXN = 5000+5, MAXM = 200000+5, MAXK = 50+5;
11 const int INF = 0x3f3f3f3f;
12 int N, M, A, B, C, S, T, K, mk;
13 int e, head[MAXN], next[MAXM], v[MAXM];
14 int d[MAXN][MAXK], w[MAXM];
15 bool inq[MAXN][MAXK];
16 queue<pair<int, int> > Q;
17 void addedge(int x, int y, int z)
18 {
19     v[e] = y; w[e] = z;
20     next[e] = head[x]; head[x] = e++;
21 }
22 void spfa(int s)
23 {
24     for (int i = 1; i <= N; i++)
25         for (int j = 1; j <= mk; j++)
26             d[i][j] = INF;
27     Q.push(make_pair(s, 0));
28     while (!Q.empty())
29     {
30         int u = Q.front().first, k = Q.front().second;
31         Q.pop();
32         inq[u][k] = 0;
33         for (int i = head[u]; i != -1; i = next[i])
34         {
35             int l = k+(k < mk ? 1 : 0);
36             if (d[u][k]+w[i] < d[v[i]][l])
37             {
38                 d[v[i]][l] = d[u][k]+w[i];
39                 if (!inq[v[i]][l])
40                 {
41                     Q.push(make_pair(v[i], l));
42                     inq[v[i]][l] = 1;
43                 }
44             }
45         }
46     }
47 }
48 void init()
49 {

```

```

50 e = 0;
51 memset(head, -1, sizeof(head));
52 }
53 int main()
54 {
55     while (scanf("%d%d", &N, &M) != EOF)
56     {
57         init();
58         for (int i = 0; i < M; i++)
59         {
60             scanf("%d%d%d", &A, &B, &C);
61             addedge(A, B, C);
62             addedge(B, A, C);
63         }
64         scanf("%d%d%d", &S, &T, &K);
65         mk = (K-1)/10+1;
66         spfa(S);
67         printf("%d\n", d[T][mk] < INF ? d[T][mk] : -1);
68     }
69     return 0;
70 }

```

找环_hdu_4337

```

01 #include<cstdio>
02 #include<cstring>
03 #include<algorithm>
04 #include<vector>
05 using namespace std;
06 const int MAXN = 150+5, MAXM = 22500+5, MAXP = 50+5;
07 int N, M, a, b;
08 int e, head[MAXN], next[MAXM], v[MAXM];
09 int mark[MAXN];
10 vector<int> vec;
11 void Init()
12 {
13     e = 0;
14     memset(head, -1, sizeof(head));
15     memset(mark, -1, sizeof(mark));
16     vec.clear();
17 }
18 void addedge(int x, int y)
19 {
20     v[e] = y;
21     next[e] = head[x]; head[x] = e++;
22 }
23 bool dfs(int u, int step = 0)
24 {
25     mark[u] = step;
26     vec.push_back(u);
27     for (int i = head[u]; i != -1; i = next[i])
28     {

```

```

29         if (mark[v[i]] == -1)
30         {
31             if (dfs(v[i], step+1))
32                 return 1;
33         }
34         else if (step-mark[v[i]]+1 == N)
35             return 1;
36     }
37     mark[u] = -1;
38     vec.pop_back();
39     return 0;
40 }
41 int main()
42 {
43     while (scanf("%d%d", &N, &M) != EOF)
44     {
45         Init();
46         for (int i = 0; i < M; i++)
47         {
48             scanf("%d%d", &a, &b);
49             addedge(a, b);
50             addedge(b, a);
51         }
52         if (dfs(1))
53         {
54             for (int i = 0; i < (int)vec.size(); i++)
55             {
56                 if (i)
57                     printf(" ");
58                 printf("%d", vec[i]);
59             }
60             printf("\n");
61         }
62         else
63             printf("no solution\n");
64     }
65     return 0;
66 }

```

最小生成树的最佳替换边_hdu_4126

```

001 /*
002 题意：给定一个图 G，有 q 次询问（相互独立），每次询问(u,v,w)，表示将<u,v>这条边的边
    权更改为 w，求此时的最小生成树的值。
003 算法：先求得最小生成树，对于每次询问(u,v,w)，分两种情况讨论：
004 1、若<u,v>是非最小生成树上的边，那么不用考虑，最小生成树仍是原来的值。
005 2、若<u,v>是最小生成树上的边，那么我们就需要在这条边所导致的两个集合中分别选出一个
    点 i,j 并且 g[i][j] 最小来替代那条被增加的边，那么反过来考虑，对于一条非最小生成树上的边
    <u,v>，它可以替代哪些边呢？就是 u->x1->x2->...->xk->v 这条路径（因为是树，所以这条路
    径唯一）上的边。那么现在要求的就是对于每条树上的边<u,v>，得到一个 best[u][v] 表示去掉它，
    最小的替代边的权值。这个可以用 dfs 来做，以每个点为起点做 dfs，遍历整个最小生成树，得到
    best[i][j]，这样复杂度就是 O(N^2) 的，然后对于每次询问就可以 O(1) 回答了。这个 dfs 的写

```

法还是有点技巧的，具体就见代码吧。

```
006 */
007 #include<stdio>
008 #include<string>
009 #include<algorithm>
010 using namespace std;
011 const int MAXN = 3000+5, MAXM = 6000+5;
012 const int INF = 0x3f3f3f3f;
013 int N, M, Q, X, Y, C;
014 int g[MAXN][MAXN], best[MAXN][MAXN], dis[MAXN], pre[MAXN];
015 int e, head[MAXN], next[MAXM], v[MAXM];
016 bool vis[MAXN];
017 void addedge(int x, int y)
018 {
019     v[e] = y;
020     next[e] = head[x]; head[x] = e++;
021 }
022 void init()
023 {
024     e = 0;
025     memset(head, -1, sizeof(head));
026     for (int i = 0; i < N; i++)
027         for (int j = 0; j < i; j++)
028             g[i][j] = g[j][i] = best[i][j] = best[j][i] = INF;
029 }
030 int prim()
031 {
032     for (int i = 0; i < N; i++)
033         vis[i] = 0, pre[i] = -1, dis[i] = INF;
034     int res = 0;
035     dis[0] = 0;
036     for (int j = 0; j < N; j++)
037     {
038         int u = -1;
039         for (int i = 0; i < N; i++)
040             if (!vis[i] && (u == -1 || dis[i] < dis[u]))
041                 u = i;
042         vis[u] = 1;
043         res += dis[u];
044         if (pre[u] != -1)
045         {
046             addedge(u, pre[u]);
047             addedge(pre[u], u);
048         }
049         for (int i = 0; i < N; i++)
050             if (!vis[i] && g[u][i] < dis[i])
051             {
052                 dis[i] = g[u][i];
053                 pre[i] = u;
054             }
055     }
056     return res;
057 }
```

```
058 int dfs(int st, int u, int fa)
059 {
060     int mini = INF;
061     for (int i = head[u]; i != -1; i = next[i]) if (v[i] != fa)
062     {
063         int cur = dfs(st, v[i], u);
064         mini = min(mini, cur);
065         best[u][v[i]] = best[v[i]][u] = min(best[u][v[i]], cur);
066     }
067     if (st != fa)
068         mini = min(mini, g[st][u]);
069     return mini;
070 }
071 int main()
072 {
073     while (scanf("%d%d", &N, &M))
074     {
075         if (!N && !M)
076             break;
077         init();
078         for (int i = 0; i < M; i++)
079         {
080             scanf("%d%d%d", &X, &Y, &C);
081             g[X][Y] = g[Y][X] = C;
082         }
083         int mst = prim();
084         for (int i = 0; i < N; i++)
085             dfs(i, i, -1);
086         scanf("%d", &Q);
087         double ans = 0;
088         for (int i = 0; i < Q; i++)
089         {
090             scanf("%d%d%d", &X, &Y, &C);
091             if (pre[X] == Y || pre[Y] == X)
092                 ans += mst - g[X][Y] + min(C, best[X][Y]);
093             else
094                 ans += mst;
095         }
096         ans /= Q;
097         printf("%.4f\n", ans);
098     }
099     return 0;
100 }
```

最小树形图_hdu_4009

```
001 /*
002 题意：有 n 个地方需要供水，每个地方都可以选择是自己挖井，还是从别的地方引水，根据方法
    不同和每个地方的坐标不同，花费也不同，现在给出每个地方的坐标，花费的计算方法，以及每个地
    方可以给哪些地方供水（即对方可以从这里引水），求给所有地方供水的最小花费。
003 思路：显然对于每个地方，只有一种供水方式就足够了，这样也能保证花费最小，而每个地方都
    可以自己挖井，所以是不可能出现无解的情况的，为了方便思考，我们引入一个虚拟点，把所有自己
```

挖井的都连到这个点，边权为挖井的花费，而如果 i 能从 j 处引水，则从 j 向 i 连边，边权为引水的花费，然后对这个有向图，以虚拟点为根，求最小树形图即可（最小树形图即为有向图的最小生成树）。

```
004 */
005 #include<cstdio>
006 #include<cstring>
007 #include<cmath>
008 #include<algorithm>
009 using namespace std;
010 const int MAXN = 1000+5, MAXM = 1001000+5;
011 const int INF = 0x3f3f3f3f;
012 int N, X, Y, Z, K, x[MAXN], y[MAXN], z[MAXN];
013 int e, u[MAXM], v[MAXM], w[MAXM];
014 int pre[MAXN], id[MAXN], vis[MAXN];
015 int in[MAXN];
016 int Directed_MST(int root,int NV,int NE) //number vertices from zero!!!
017 {
018     int res = 0;
019     for (;;)
020     {
021         //1.找最小入边
022         for (int i = 0; i < NV; i++)
023             in[i] = INF, id[i] = -1, vis[i] = -1;
024         for (int i = 0; i < NE; i++)
025         {
026             int s = u[i], t = v[i];
027             if (w[i] < in[t] && s != t)
028             {
029                 pre[t] = s;
030                 in[t] = w[i];
031             }
032         }
033         for (int i = 0; i < NV; i++)
034         {
035             if (i == root)
036                 continue;
037             if (in[i] == INF)
038                 return -1; //除了跟以外有点没有入边,则根无法到达它
039         }
040         //2.找环
041         int cntnode = 0;
042         in[root] = 0;
043         for (int i = 0; i < NV; i++)
044             //标记每个环
045             res += in[i];
046         int t = i;
047         for (; vis[t] != i && id[t] == -1 && t != root; t = pre[t])
048             vis[t] = i;
049         if (t != root && id[t] == -1)
050         {
051             for (int s = pre[t] ; s != t ; s = pre[s])
052                 id[s] = cntnode;
053             id[t] = cntnode++;
054         }
055     }
```

```

055     }
056     if (!cntnode)
057         break; //无环
058     for (int i = 0; i < NV; i++)
059         if (id[i] == -1)
060             id[i] = cntnode++;
061     //3.缩点,重新标记
062     for (int i = 0; i < NE; i++)
063     {
064         int t = v[i];
065         u[i] = id[u[i]];
066         v[i] = id[v[i]];
067         if (u[i] != v[i])
068             w[i] -= in[t];
069     }
070     NV = cntnode;
071     root = id[root];
072 }
073 return res;
074 }
075 void addedge(int x, int y, int z)
076 {
077     u[e] = x; v[e] = y; w[e] = z;
078     e++;
079 }
080 int main()
081 {
082     while (scanf("%d%d%d", &N, &X, &Y, &Z))
083     {
084         if (!N && !X && !Y && !Z)
085             break;
086         e = 0;
087         int root = 0;
088         for (int i = 1; i <= N; i++)
089         {
090             scanf("%d%d", &x[i], &y[i], &z[i]);
091             addedge(root, i, z[i]*X);
092         }
093         for (int i = 1; i <= N; i++)
094         {
095             scanf("%d", &K);
096             for (int k = 1, j; k <= K; k++)
097             {
098                 scanf("%d", &j);
099                 if (z[i] < z[j])
100                     addedge(i, j, (abs(x[i]-x[j])+abs(y[i]-y[j])+abs(z[i]-z[j]))*
Y+Z);
101             else
102                 addedge(i, j, (abs(x[i]-x[j])+abs(y[i]-y[j])+abs(z[i]-z[j]))*
Y);
103             }
104         }
105         printf("%d\n", Directed_MST(root, N+1, e));

```

```

106 }
107 return 0;
108 }

```

Network

最大流 ISAP_hdu_3879

```

01 #include<cstdio>
02 #include<cstring>
03 #include<algorithm>
04 using namespace std;
05 const int MAXN = 55000+5, MAXM = 155000*2+5;
06 const int INF = 0x3f3f3f3f;
07 int N, M;
08 int n, s, t;
09 int e, v[MAXN], next[MAXN], head[MAXN];
10 int cap[MAXN];
11 int h[MAXN], gap[MAXN];
12 void init()
13 {
14     e = 0;
15     memset(head, -1, sizeof(head));
16     memset(gap, 0, sizeof(gap));
17     memset(h, 0, sizeof(h));
18 }
19 void addedge(int x, int y, int c)
20 {
21     v[e] = y; cap[e] = c;
22     next[e] = head[x]; head[x] = e++;
23     v[e] = x; cap[e] = 0;
24     next[e] = head[y]; head[y] = e++;
25 }
26 int sap(int u, int f)
27 {
28     if (u == t)
29         return f;
30     int minh = n-1, rf = f;
31     for (int i = head[u]; i != -1; i = next[i]) if (cap[i])
32     {
33         if (h[v[i]]+1 == h[u])
34         {
35             int cf = sap(v[i], min(cap[i], rf));
36             cap[i] -= cf;
37             cap[i^1] += cf;
38             rf -= cf;
39             if (h[s] >= n)
40                 return f-rf;
41             if (!rf)

```

```

42         break;
43     }
44     minh = min(minh, h[v[i]]);
45 }
46 if (rf == f)
47 {
48     gap[h[u]]--;
49     if (!gap[h[u]])
50         h[s] = n;
51     h[u] = minh+1;
52     gap[h[u]]++;
53 }
54 return f-rf;
55 }
56 int maxflow()
57 {
58     int res = 0;
59     gap[0] = n;
60     while (h[s] < n)
61         res += sap(s, INF);
62     return res;
63 }
64 int main()
65 {
66     freopen("input.txt", "r", stdin);
67     // freopen("output.txt", "w", stdout);
68     while (scanf("%d%d", &N, &M) != EOF)
69     {
70         init();
71         n = N+M+2; s = N+M+1; t = s+1;
72         for (int i = 1; i <= N; i++)
73         {
74             int P;
75             scanf("%d", &P);
76             addedge(i, t, P);
77         }
78         int tp = 0;
79         for (int i = 1; i <= M; i++)
80         {
81             int x, y, z;
82             scanf("%d%d%d", &x, &y, &z);
83             tp += z;
84             addedge(s, N+i, z);
85             addedge(N+i, x, INF);
86             addedge(N+i, y, INF);
87         }
88         int f = maxflow();
89         printf("%d\n", tp-f);
90     }
91     return 0;
92 }

```


最大流-邻接表

```

01 #include<cstdio>
02 #include<cstring>
03 #include<algorithm>
04 #include<queue>
05 using namespace std;
06 const int MAXN = 1000+5, MAXM = 1000+5;
07 const int INF = 0x3f3f3f3f;
08 int e, s, t, n;
09 int v[MAXN], next[MAXN], head[MAXN];
10 int cap[MAXN], a[MAXN], f;
11 int pv[MAXN], pe[MAXN];
12 queue<int> Q;
13 void addedge(int u_, int v_, int c_)
14 {
15     v[e] = v_; cap[e] = c_;
16     next[e] = head[u_]; head[u_] = e;
17     e++;
18     v[e] = u_; cap[e] = 0;
19     next[e] = head[v_]; head[v_] = e;
20     e++;
21 }
22 void maxflow()
23 {
24     f = 0;
25     for (;;)
26     {
27         memset(a, 0, sizeof(a));
28         a[s] = INF;
29         Q.push(s);
30         while (!Q.empty())
31         {
32             int u = Q.front(); Q.pop();
33             for (int e = head[u]; e != -1; e = next[e])
34                 if (!a[v[e]] && cap[e])
35                 {
36                     Q.push(v[e]);
37                     a[v[e]] = min(a[u], cap[e]);
38                     pv[v[e]] = u; pe[v[e]] = e;
39                 }
40         }
41         if (!a[t]) break;
42         for (int v = t; v != s; v = pv[v])
43         {
44             cap[pe[v]] -= a[t];
45             cap[pe[v]^1] += a[t];
46         }
47         f += a[t];
48     }
49 }
50 int main()
51 {

```

```

52 // freopen("input.txt", "r", stdin);
53 // freopen("output.txt", "w", stdout);
54 memset(cap, 0, sizeof(cap));
55 memset(head, -1, sizeof(head));
56 e = 0;
57
58 return 0;
59 }

```

最大流-邻接矩阵

```

01 #include<cstdio>
02 #include<cstring>
03 #include<algorithm>
04 #include<queue>
05 using namespace std;
06 const int MAXN = 1000+5;
07 const int INF = 0x3f3f3f3f;
08 int s, t, n;
09 int p[MAXN];
10 int cap[MAXN][MAXN], flow[MAXN][MAXN], a[MAXN], f;
11 queue<int> Q;
12 void addedge(int u_, int v_, int c_)
13 {
14     cap[u_][v_] = c_;
15 }
16 void maxflow()
17 {
18     f = 0;
19     memset(flow, 0, sizeof(flow));
20     for (;;)
21     {
22         memset(a, 0, sizeof(a));
23         a[s] = INF;
24         Q.push(s);
25         while (!Q.empty())
26         {
27             int u = Q.front(); Q.pop();
28             for (int v = 1; v <= n; v++)
29                 if (!a[v] && cap[u][v] > flow[u][v])
30                 {
31                     p[v] = u; Q.push(v);
32                     a[v] = min(a[u], cap[u][v]-flow[u][v]);
33                 }
34         }
35         if (a[t] == 0) break;
36         for (int v = t; v != s; v = p[v])
37         {
38             flow[p[v]][v] += a[t];
39             flow[v][p[v]] -= a[t];
40         }
41         f += a[t];

```

```

42 }
43 }
44 int main()
45 {
46 // freopen("input.txt", "r", stdin);
47 // freopen("output.txt", "w", stdout);
48 memset(cap, 0, sizeof(cap));
49
50 return 0;
51 }

```

最小费用最大流-邻接表

```

01 #include<cstdio>
02 #include<cstring>
03 #include<algorithm>
04 #include<queue>
05 using namespace std;
06 const int MAXN = 1000+5, MAXM = 1000+5;
07 const int INF = 0x3f3f3f3f;
08 int e, s, t, n;
09 int v[MAXN], next[MAXN], head[MAXN];
10 int cap[MAXN], f;
11 int cost[MAXN], d[MAXN], c;
12 int pv[MAXN], pe[MAXN];
13 bool inq[MAXN];
14 queue<int> Q;
15 void addedge(int u_, int v_, int c_, int w_)
16 {
17 v[e] = v_; cap[e] = c_; cost[e] = w_;
18 next[e] = head[u_]; head[u_] = e;
19 e++;
20 v[e] = u_; cap[e] = 0; cost[e] = -w_;
21 next[e] = head[v_]; head[v_] = e;
22 e++;
23 }
24 void mincostflow()
25 {
26 f = 0; c = 0;
27 for (;;)
28 {
29 memset(inq, 0, sizeof(inq));
30 for (int i = 1; i <= n; i++)
31 d[i] = (i == s ? 0 : INF);
32 Q.push(s); inq[s] = 1;
33 while (!Q.empty())
34 {
35 int u = Q.front(); Q.pop();
36 inq[u] = 0;
37 for (int e = head[u]; e != -1; e = next[e])
38 if (cap[e] && d[v[e]] > d[u]+cost[e])
39 {

```

```

40 d[v[e]] = d[u]+cost[e];
41 if (!inq[v[e]])
42 Q.push(v[e]), inq[v[e]] = 1;
43 pv[v[e]] = u; pe[v[e]] = e;
44 }
45 }
46 if (d[t] == INF) break;
47 int a = INF;
48 for (int v = t; v != s; v = pv[v])
49 a = min(a, cap[pe[v]]);
50 for (int v = t; v != s; v = pv[v])
51 {
52 cap[pe[v]] -= a;
53 cap[pe[v]^1] += a;
54 }
55 f += a;
56 c += d[t]*a;
57 }
58 }
59 int main()
60 {
61 // freopen("input.txt", "r", stdin);
62 // freopen("output.txt", "w", stdout);
63 memset(cap, 0, sizeof(cap));
64 memset(cost, 0, sizeof(cost));
65 memset(head, -1, sizeof(head));
66 e = 0;
67
68 return 0;
69 }

```

最小费用最大流-邻接矩阵

```

01 #include<cstdio>
02 #include<cstring>
03 #include<algorithm>
04 #include<queue>
05 using namespace std;
06 const int MAXN = 1000+5;
07 const int INF = 0x3f3f3f3f;
08 int s, t, n;
09 int cost[MAXN][MAXN], d[MAXN], c;
10 int cap[MAXN][MAXN], flow[MAXN][MAXN], f;
11 int p[MAXN];
12 bool inq[MAXN];
13 queue<int> Q;
14 void addedge(int u_, int v_, int c_, int w_)
15 {
16 cap[u_][v_] = c_;
17 cost[u_][v_] = w_; cost[v_][u_] = -w_;
18 }
19 void mincostflow()

```

```

20 {
21     f = 0, c = 0;
22     memset(flow, 0, sizeof(flow));
23     for(;;)
24     {
25         for(int i = 1; i <= n; i++)
26             d[i] = (i == s ? 0 : INF);
27         memset(inq, 0, sizeof(inq));
28         Q.push(s); inq[s] = 1;
29         while(!Q.empty())
30         {
31             int u = Q.front(); Q.pop();
32             inq[u] = 0;
33             for(int v = 1; v <= n; v++)
34                 if(cap[u][v] > flow[u][v] && d[v] > d[u]+cost[u][v])
35                 {
36                     d[v] = d[u]+cost[u][v];
37                     if(!inq[v])
38                         Q.push(v), inq[v] = 1;
39                     p[v] = u;
40                 }
41         }
42         if (d[t] == INF) break;
43         int a = INF;
44         for(int v = t; v != s; v = p[v])
45             a = min(a, cap[p[v]][v]-flow[p[v]][v]);
46         for(int v = t; v != s; v = p[v])
47         {
48             flow[p[v]][v] += a;
49             flow[v][p[v]] -= a;
50         }
51         c += d[t]*a;
52         f += a;
53     }
54 }
55 int main()
56 {
57     freopen("input.txt", "r", stdin);
58     freopen("output.txt", "w", stdout);
59     memset(cap, 0, sizeof(cap));
60     memset(cost, 0, sizeof(cost));
61
62     return 0;
63 }

```

Number

组合数 C(N, R)

```

01 int com(int n, int r)
02 { // return C(n, r)
03     if (n-r > r) r = n-r; // C(n, r) = C(n, n-r)
04     int s = 1;
05     for (int i = 0, j = 1; i < r; i++)
06     {
07         s *= (n-i);
08         for(; j <= r && s%j == 0; j++)
09             s /= j;
10     }
11     return s;
12 }

```

Structure

AC 自动机

AC 自动机_hdu_2222

```

001 /*
002 网络流上流传最广的 AC 自动机模板题，问你目标串中出现了几个模式串
003 如果一个结点是单词末尾的话 out 标记为 true, 在 search 的时候对于每个结点都向 fail 指针找，找到 out 为 true 的就将其标记为 false, 且 ans+=out
004 */
005 #include<cstdio>
006 #include<cstring>
007 #include<algorithm>
008 #include<queue>
009 using namespace std;
010 const int MAXN = 1000000+5, MAXM = 50+5;
011 const int MAX_NODE = 500000+5, MAX_CHD = 26;
012 int T, N;
013 int chd[MAX_NODE][MAX_CHD], fail[MAX_NODE], out[MAX_NODE];
014 int ID[1<<8], nv;
015 char key[MAXN], des[MAXN];
016 queue<int> Q;
017 namespace AC_Automaton
018 {
019     void Initialize()
020     {
021         fail[0] = 0;

```

```

022     for (int i = 0; i < MAX_CHD; i++)
023         ID[i+'a'] = i;
024 }
025 void Reset()
026 {
027     memset(chd[0], 0, sizeof(chd[0]));
028     nv = 1;
029 }
030 void Insert(char *pat)
031 {
032     int u = 0;
033     for (int i = 0; pat[i]; i++)
034     {
035         int c = ID[pat[i]];
036         if (!chd[u][c])
037         {
038             memset(chd[nv], 0, sizeof(chd[nv]));
039             out[nv] = 0;
040             chd[u][c] = nv++;
041         }
042         u = chd[u][c];
043     }
044     out[u]++;
045 }
046 void Construct()
047 {
048     for (int i = 0; i < MAX_CHD; i++)
049         if (chd[0][i])
050         {
051             fail[chd[0][i]] = 0;
052             Q.push(chd[0][i]);
053         }
054     while (!Q.empty())
055     {
056         int u = Q.front(); Q.pop();
057         for (int i = 0; i < MAX_CHD; i++)
058         {
059             int v = chd[u][i];
060             if (v)
061             {
062                 Q.push(v);
063                 fail[v] = chd[fail[u]][i];
064             }
065             else
066                 chd[u][i] = chd[fail[u]][i];
067         }
068     }
069 }
070 }
071 int main()
072 {
073     AC_Automaton::Initialize();
074     scanf("%d", &T);

```

```

075     while (T--)
076     {
077         scanf("%d", &N);
078         AC_Automaton::Reset();
079         for (int i = 0; i < N; i++)
080         {
081             scanf("%s", key);
082             AC_Automaton::Insert(key);
083         }
084         AC_Automaton::Construct();
085         scanf("%s", des);
086         int ans = 0;
087         for (int i = 0, u = 0; des[i]; i++)
088         {
089             u = chd[u][ID[des[i]]];
090             for (int t = u; t; )
091             {
092                 ans += out[t];
093                 out[t] = 0;
094                 t = fail[t];
095             }
096         }
097         printf("%d\n", ans);
098     }
099     return 0;
100 }

```

AC 自动机+DP_hdu_2825

```

001 /*
002 求长度为 n 的字符串中包含至少 k 个给出的关键字的字符串的个数，结果模 MOD。
003 */
004 #include<cstdio>
005 #include<cstring>
006 #include<algorithm>
007 #include<queue>
008 using namespace std;
009
010 //MAX_NODE = StringNumber*StringLength
011 const int MAX_NODE = 100+5;
012 //字符集大小，一般字符形式的题 26 个
013 const int MAX_CHD = 26;
014 //每个节点的儿子，即当前节点的状态转移
015 int chd[MAX_NODE][MAX_CHD];
016 //记录题目给的关键数据(点的权值)
017 int out[MAX_NODE];
018 //传说中的 fail 指针
019 int fail[MAX_NODE];
020 //字母对应的 ID
021 int ID[1<<8];
022 //已使用节点个数
023 int nv;

```

```

024 //队列,用于广度优先计算 fail 指针
025 queue<int> Q;
026
027 //特定题目需要
028 const int MAXN = 25+5;
029 const int MOD = 20090717;
030 int N, M, K, d[2][MAX_NODE][1<<10];
031
032 namespace AC_Automaton
033 {
034     //初始化,计算字母对应的儿子 ID,如:'a'→0 ... 'z'→25
035     void Initialize()
036     {
037         fail[0] = 0;
038         for (int i = 0; i < MAX_CHD; i++)
039             ID[i+'a'] = i;
040     }
041     //重新建树需先 Reset
042     void Reset()
043     {
044         memset(chd[0], 0, sizeof(chd[0]));
045         nv = 1;
046     }
047     //将权值为 key 的字符串 a 插入到 trie 中
048     void Insert(char *pat, int key)
049     {
050         int u = 0;
051         for (int i = 0; pat[i]; i++)
052         {
053             int c = ID[pat[i]];
054             if (!chd[u][c])
055             {
056                 memset(chd[nv], 0, sizeof(chd[nv]));
057                 out[nv] = 0;
058                 chd[u][c] = nv++;
059             }
060             u = chd[u][c];
061         }
062         out[u] = key;
063     }
064     //建立 AC 自动机,确定每个节点的权值以及状态转移
065     void Construct()
066     {
067         for (int i = 0; i < MAX_CHD; i++)
068             if (chd[0][i])
069             {
070                 fail[chd[0][i]] = 0;
071                 Q.push(chd[0][i]);
072             }
073         while (!Q.empty())
074         {
075             int u = Q.front(); Q.pop();
076             for (int i = 0; i < MAX_CHD; i++)

```

```

077         {
078             int &v = chd[u][i];
079             if (v)
080             {
081                 Q.push(v);
082                 fail[v] = chd[fail[u]][i];
083                 //以下一行代码要根据题目所给 out 的含义来写
084                 out[v] |= out[fail[v]];
085             }
086             else
087                 v = chd[fail[u]][i];
088         }
089     }
090 }
091 }
092
093 //解题
094 int solve()
095 {
096     int tot = (1<<M)-1, ans = 0, s = 0, t = 1;
097     memset(d[t], 0, sizeof(d[t]));
098     d[t][0][0] = 1;
099     for (int i = 0; i < N; i++)
100     {
101         swap(s, t);
102         memset(d[t], 0, sizeof(d[t]));
103         for (int u = 0; u < nv; u++)
104             for (int a = 0; a <= tot; a++) if (d[s][u][a])
105                 for (int k = 0; k < MAX_CHD; k++)
106                 {
107                     int v = chd[u][k], b = (a|out[v]);
108                     d[t][v][b] = (d[t][v][b]+d[s][u][a])%MOD;
109                 }
110     }
111     for (int a = 0; a <= tot; a++)
112     {
113         int cnt = 0;
114         for (int i = 0; i < M; i++)
115             if (a&(1<<i))
116                 cnt++;
117         if (cnt >= K)
118         {
119             for (int u = 0; u < nv; u++)
120                 ans = (ans+d[t][u][a])%MOD;
121         }
122     }
123     return ans;
124 }
125
126 int main()
127 {
128     AC_Automaton::Initialize();
129     while (scanf("%d%d%d", &N, &M, &K) != EOF)

```

```

130 {
131     if (!N && !M && !K)
132         break;
133     AC_Automaton::Reset();
134     for (int i = 0; i < M; i++)
135     {
136         char temp[11];
137         scanf("%s", temp);
138         AC_Automaton::Insert(temp, 1<<i);
139     }
140     AC_Automaton::Construct();
141     printf("%d\n", solve());
142 }
143 return 0;
144 }

```

AC 自动机+概率 DP_hdu_3689

```

001 /*
002 字符集中有一些字符，给出每个字符的出现概率（它们的和保证为 1），再给出一个串 s，问任给
003 一个长度为 N 的字符串 A（只能包含字符集中的字符），使得 S 是 A 的子串的概率。
004 */
005 #include<cstdio>
006 #include<cstring>
007 #include<algorithm>
008 #include<queue>
009 using namespace std;
010 const int MAXN = 1000+5, MAXM = 10+5;
011 const int INF = 0x3f3f3f3f;
012 const int MAX_NODE = MAXN, MAX_CHD = 26;
013 int chd[MAX_NODE][MAX_CHD], fail[MAX_NODE], out[MAX_NODE];
014 int ID[1<<8], nv;
015 double P[MAX_CHD], d[MAXN][MAX_NODE];
016 char ch[5], word[MAXM];
017 queue<int> Q;
018 namespace AC_Automaton
019 {
020     void Initialize()
021     {
022         fail[0] = 0;
023         for (int i = 0; i < MAX_CHD; i++)
024             ID[i+'a'] = i;
025     }
026     void Reset()
027     {
028         memset(chd[0], 0, sizeof(chd[0]));
029         nv = 1;
030     }
031     void Insert(char *pat)
032     {
033         int u = 0;

```

```

034     for (int i = 0; pat[i]; i++)
035     {
036         int c = ID[pat[i]];
037         if (!chd[u][c])
038         {
039             memset(chd[nv], 0, sizeof(chd[nv]));
040             out[nv] = 0;
041             chd[u][c] = nv++;
042         }
043         u = chd[u][c];
044     }
045     out[u]++;
046 }
047 void Construct()
048 {
049     for (int i = 0; i < MAX_CHD; i++)
050         if (chd[0][i])
051         {
052             fail[chd[0][i]] = 0;
053             Q.push(chd[0][i]);
054         }
055     while (!Q.empty())
056     {
057         int u = Q.front(); Q.pop();
058         for (int i = 0; i < MAX_CHD; i++)
059         {
060             int &v = chd[u][i];
061             if (v)
062             {
063                 Q.push(v);
064                 fail[v] = chd[fail[u]][i];
065             }
066             else
067                 v = chd[fail[u]][i];
068         }
069     }
070 }
071 }
072 int main()
073 {
074     AC_Automaton::Initialize();
075     while (scanf("%d%d", &N, &M))
076     {
077         if (!N && !M)
078             break;
079         memset(P, 0, sizeof(P));
080         memset(d, 0, sizeof(d));
081         AC_Automaton::Reset();
082         for (int i = 0; i < N; i++)
083         {
084             scanf("%s", ch);
085             scanf("%lf", &P[ID[ch[0]]]);
086         }

```

```

087     scanf("%s", word);
088     AC_Automaton::Insert(word);
089     AC_Automaton::Construct();
090     d[0][0] = 1;
091     for (int i = 0; i < M; i++)
092         for (int u = 0; u < nv; u++) if (d[i][u] && !out[u])
093             for (int j = 0; j < MAX_CHD; j++)
094                 d[i+1][chd[u][j]] += d[i][u]*P[j];
095     int len = strlen(word);
096     double ans = 0;
097     for (int i = len; i <= M; i++)
098         ans += d[i][len];
099     printf("%.2lf%s\n", ans*100, "\\");
100 }
101 return 0;
102 }

```

AC 自动机+矩阵_poj_2778

```

001 /*
002 问你长度为 N 的串中不包含模式串的串有几个
003 n 属于 1 ~ 2000000000 看到这个数据范围我们就应该敏感的想到这是矩阵~
004 最多 100 个结点, 先建好所有结点(不包括模式串结尾的和 fail 指向结尾的结点, 所以其实最多只有 90 个有效结点)之间的转化关系, 然后二分矩阵乘法, 复杂度  $O(100^3 * \log(2000000000))$ 
005 */
006 #include<cstdio>
007 #include<cstring>
008 #include<algorithm>
009 #include<queue>
010 using namespace std;
011 const int MAXM = 10+5;
012 const int MAX_NODE = 100+5, MAX_CHD = 4;
013 const long long MOD = 100000;
014 typedef long long MAT[MAX_NODE][MAX_NODE];
015 MAT g, G;
016 int M, N;
017 int chd[MAX_NODE][MAX_CHD], fail[MAX_NODE], ID[1<<8], nv;
018 bool out[MAX_NODE];
019 char DNA[MAXM];
020 queue<int> Q;
021 namespace AC_Automaton
022 {
023     void Initialize()
024     {
025         fail[0] = 0;
026         ID['A'] = 0; ID['C'] = 1; ID['T'] = 2; ID['G'] = 3;
027     }
028     void Reset()
029     {
030         memset(chd[0], 0, sizeof(chd[0]));
031         nv = 1;
032     }

```

```

033 void Insert(char *pat)
034 {
035     int u = 0;
036     for (int i = 0; pat[i]; i++)
037     {
038         int c = ID[pat[i]];
039         if (!chd[u][c])
040         {
041             memset(chd[nv], 0, sizeof(chd[nv]));
042             out[nv] = 0;
043             chd[u][c] = nv++;
044         }
045         u = chd[u][c];
046     }
047     out[u] = 1;
048 }
049 void Construct()
050 {
051     for (int i = 0; i < MAX_CHD; i++)
052         if (chd[0][i])
053         {
054             fail[chd[0][i]] = 0;
055             Q.push(chd[0][i]);
056         }
057     while (!Q.empty())
058     {
059         int u = Q.front(); Q.pop();
060         for (int i = 0; i < MAX_CHD; i++)
061         {
062             int &v = chd[u][i];
063             if (v)
064             {
065                 Q.push(v);
066                 fail[v] = chd[fail[u]][i];
067                 out[v] |= out[fail[v]];
068             }
069             else
070                 v = chd[fail[u]][i];
071         }
072     }
073 }
074 }
075 namespace Matrix
076 {
077     void Copy(int size, MAT x, MAT y)
078     {
079         for (int i = 0; i < size; i++)
080             for (int j = 0; j < size; j++)
081                 y[i][j] = x[i][j];
082     }
083     void Mutiply(int size, MAT x, MAT y, MAT z)
084     {
085         MAT tx, ty;

```

```

086 Copy(size, x, tx);
087 Copy(size, y, ty);
088 for (int i = 0; i < size; i++)
089     for (int j = 0; j < size; j++)
090     {
091         z[i][j] = 0;
092         for (int k = 0; k < size; ++k)
093             z[i][j] = (z[i][j]+tx[i][k]*ty[k][j])%MOD;
094     }
095 }
096 void Power(int size, MAT x, int n, MAT y)
097 {
098     MAT tx, r;
099     Copy(size, x, tx);
100     for (int i = 0; i < size; i++)
101         for (int j = 0; j < size; j++)
102             r[i][j] = (i == j ? 1 : 0);
103     while (n)
104     {
105         if (n&1)
106             Mutiply(size, r, tx, r);
107         n >>= 1;
108         if (!n)
109             break;
110         Mutiply(size, tx, tx, tx);
111     }
112     Copy(size, r, y);
113 }
114 }
115 int main()
116 {
117     AC_Automaton::Initialize();
118     memset(g, 0, sizeof(g));
119     AC_Automaton::Reset();
120     scanf("%d%d", &M, &N);
121     for (int i = 0; i < M; i++)
122     {
123         scanf("%s", DNA);
124         AC_Automaton::Insert(DNA);
125     }
126     AC_Automaton::Construct();
127     for (int u = 0; u < nv; u++) if (!out[u])
128         for (int k = 0; k < MAX_CHD; k++) if (!out[chd[u][k]])
129             g[u][chd[u][k]]++;
130     Matrix::Power(nv, g, N, G);
131     long long ans = 0;
132     for (int i = 0; i < nv; i++)
133         ans = (ans+G[0][i])%MOD;
134     printf("%lld\n", ans);
135     return 0;
136 }

```

DP

离散 DP_hdu_4028

```

01 /*
02 题意：给你 n 个钟的指针，第 i 个指针转一圈的时间是 i 单位，问你从 n 个钟任选一些指针使得，
全部指针第一次回到原来的位置是经过的时间大于等于 m，求又多少种选法。
03 思路：显然时间是你选的指针的最小公倍数，但是好大，dp 无从下手。看完神牛的题解才知道有
一种 dp 叫做离散 dp，就是直接保存有用的状态就好了，其他的不用，这样空间就可以满足了，因为
其实状态数很少。状态设定很简单：dp[i][j]：i 表示以 i 指针结尾，最小公倍数 (lcm) 为 j 的方
案数。转移也很简单就是 dp[i][j]=dp[i][j]+dp[i-1][j]；离散用了 map，STL 太强了，只能
这么感慨，map 要注意 lcm 的转移；还有初始状态为 dp[i][i]=1；要在更新这个状态的时候加进去：
04 */
05 #include<cstdio>
06 #include<cstring>
07 #include<algorithm>
08 #include<map>
09 using namespace std;
10 const int MAX = 40, MAXN = MAX+5;
11 const int INF = 0x3f3f3f3f;
12 int T, N;
13 long long M;
14 struct cmp
15 {
16     bool operator()(const long long a, const long long b)
17     {
18         return a > b;
19     }
20 };
21 map<long long, long long, cmp> d[MAXN];
22 long long gcd(long long x, long long y)
23 {
24     return !y ? x : gcd(y, x%y);
25 }
26 long long lcm(long long x, long long y)
27 {
28     return x/gcd(x, y)*y;
29 }
30 int main()
31 {
32     scanf("%d", &T);
33     for (int i = 1; i <= MAX; i++)
34     {
35         d[i] = d[i-1];
36         d[i][i]++;
37         map<long long, long long, cmp>::iterator p = d[i-1].begin();
38         for (; p != d[i-1].end(); p++)
39             d[i][lcm(p->first, i)] += p->second;
40     }
41     for (int cas = 1; cas <= T; cas++)
42     {
43         scanf("%d%I64d", &N, &M);

```



```

44     long long ans = 0;
45     map<long long, long long, cmp>::iterator p = d[N].begin();
46     for (; p != d[N].end() && p->first >= M; p++)
47         ans += p->second;
48     printf("Case #%d: %I64d\n", cas, ans);
49 }
50 return 0;
51 }

```

区间 DP_hdu_4293_1

```

01 /*
02 题意：每个区间有权值，给若干区间，求最大收益。
03 思路：d[i]表示长度为 i 且包含以 i 结尾的区间时最大的人数。
04 */
05 #include<cstdio>
06 #include<cstring>
07 #include<algorithm>
08 using namespace std;
09 const int MAXN = 500+5;
10 const int INF = 0x3f3f3f3f;
11 int N, a, b, A[MAXN], B[MAXN], r[MAXN];
12 int mp[MAXN][MAXN], num[MAXN], d[MAXN];
13 bool cmp(const int a, const int b)
14 {
15     return B[a] < B[b];
16 }
17 int main()
18 {
19     while (scanf("%d", &N) != EOF)
20     {
21         memset(mp, 0, sizeof(mp));
22         memset(num, 0, sizeof(num));
23         memset(d, 0, sizeof(d));
24         int n = 0, ans = 0;
25         for (int i = 1; i <= N; i++)
26         {
27             scanf("%d%d", &a, &b);
28             if (a+b >= N)
29                 continue;
30             int &m = mp[a+1][N-b];
31             if (!m)
32             {
33                 m = ++n;
34                 A[n] = a+1;
35                 B[n] = N-b;
36                 r[n] = n;
37             }
38             num[m] = min(num[m]+1, N-a-b);
39         }
40         sort(r+1, r+1+n, cmp);
41         for (int i = 1; i <= n; i++)

```

```

42         for (int j = 0; j < A[r[i]]; j++)
43             d[B[r[i]]] = max(d[B[r[i]]], d[j]+num[r[i]]);
44         for (int i = 1; i <= N; i++)
45             ans = max(ans, d[i]);
46         printf("%d\n", ans);
47     }
48     return 0;
49 }

```

树形背包 DP_hdu_4276

```

01 /*
02 题意：一个有 N 个节点的树形的地图，知道了每条边经过所需要的时间，现在给出时间 T，问能
    不能在 T 时间内从 1 号节点到 N 节点。每个节点都有相对应的价值，而且每个价值只能被取一次，问
    如果可以从 1 号节点走到 n 号节点的话，最多可以取到的最大价值为多少。
03 分析：先求出从 1 号节点到 n 号节点的最短路，如果花费大于时间 T，则直接输出不符合，将最
    短路上的权值全部赋值为 0，在总时间 T 上减去最短路的长度，表示最短路已经走过，对其它点进行
    树形背包求解，需要注意的是如果不是最短路上的边都要走两次，即走过去还要再走回来，状态转移
    方程：dp[i][j]=max(dp[i][j],dp[i][k]+dp[i][j-2*val-k])
04 */
05 #include<cstdio>
06 #include<cstring>
07 #include<algorithm>
08 using namespace std;
09 const int MAXN = 100+5, MAXM = 500+5;
10 int N, T, a, b, t, A[MAXN];
11 int e, head[MAXN], next[MAXM], v[MAXM], w[MAXM];
12 int fa[MAXN], d[MAXN][MAXM];
13 void addedge(int x, int y, int z)
14 {
15     v[e] = y; w[e] = z;
16     next[e] = head[x]; head[x] = e++;
17 }
18 void mark(int u)
19 {
20     for (int i = head[u]; i != -1; i = next[i]) if (v[i] != fa[u])
21     {
22         fa[v[i]] = u;
23         mark(v[i]);
24     }
25 }
26 void dfs(int u, int C)
27 {
28     fill(d[u], d[u]+1+C, A[u]);
29     for (int i = head[u]; i != -1; i = next[i]) if (v[i] != fa[u])
30     {
31         int cost = w[i]*2;
32         if (cost <= C)
33         {
34             dfs(v[i], C-cost);
35             for (int j = C; j >= 0; j--)
36                 for (int k = 0; k <= j-cost; k++)

```

```

37     d[u][j] = max(d[u][j], d[u][j-k-cost]+d[v[i]][k]);
38 }
39 }
40 }
41 int main()
42 {
43     while (scanf("%d%d", &N, &T) != EOF)
44     {
45         e = 0;
46         memset(head, -1, sizeof(head));
47         for (int i = 1; i < N; i++)
48         {
49             scanf("%d%d%d", &a, &b, &t);
50             addedge(a, b, t);
51             addedge(b, a, t);
52         }
53         for (int i = 1; i <= N; i++)
54             scanf("%d", &A[i]);
55         int ans = 0;
56         mark(1);
57         for (int u = N; ; )
58         {
59             ans += A[u];
60             A[u] = 0;
61             if (u == 1)
62                 break;
63             for (int i = head[u]; i != -1; i = next[i]) if (v[i] == fa[u])
64             {
65                 u = v[i];
66                 T -= w[i];
67                 w[i] = 0;
68                 w[i^1] = 0;
69                 break;
70             }
71         }
72         if (T < 0)
73             printf("Human beings die in pursuit of wealth, and birds die in pursuit
of food!\n");
74         else
75         {
76             dfs(1, T);
77             ans += d[1][T];
78             printf("%d\n", ans);
79         }
80     }
81     return 0;
82 }

```

KMP

扩展 KMP_hdu_4300

```

01 /*
02 这道题问的就是将 1 个串如何变为 stringA+stringB 的形式，使得 stringA 是 stringB 经
过映射得到相同的串。映射那步其实没有什么价值，假设 str 为原串 s 经过映射后得到的串，我们可
以以 str 为模式串，以 s 为原串做一次扩展 KMP，得到 extend 数组，extend[i] 表示原串以第 i
开始与模式串的前缀的最长匹配。经过 O(n) 的枚举，我们可以得到，若 extend[i]+i=len 且
i>=extend[i] 时，表示 stringB 即为该点之前的串，stringA 即为该点之前的 str 串，最后输出
即可。
03 */
04 #include<cstdio>
05 #include<cstring>
06 #include<algorithm>
07 using namespace std;
08 const int MAXN = 100000+5, MAXM = 50+5;
09 const int INF = 0x3f3f3f3f;
10 int T, extend[MAXN], next[MAXN];
11 char S[MAXN], tex1[MAXN], tex2[MAXN], match[1<<8];
12 void get_next(char *pat)
13 {
14     int len2 = strlen(pat), k = 0;
15     next[0] = len2;
16     while (k+1 < len2 && pat[k] == pat[k+1])
17         k++;
18     next[1] = k;
19     for(int id = 1, i = 2; i < len2; i++)
20     {
21         int u = i-id;
22         if (next[u]+i >= next[id]+id)
23         {
24             int j = next[id]+id-i;
25             if (j < 0)
26                 j = 0;
27             while (j+i < len2 && pat[j] == pat[j+i])
28                 j++;
29             next[i] = j;
30             id = i;
31         }
32         else
33             next[i] = next[u];
34     }
35 }
36 void ext_kmp(char *str, char *pat)
37 {
38     get_next(pat);
39     int len1 = strlen(str), len2 = strlen(pat), k = 0;
40     while (k < len1 && k < len2 && str[k] == pat[k])
41         k++;
42     extend[0] = k;
43     for (int id = 0, i = 1; i < len1; i++)

```

```

44 {
45     int u = i-id;
46     if (i+next[u] < extend[id]+id)
47         extend[i] = next[u];
48     else
49     {
50         int j = extend[id]+id-i;
51         if (j < 0)
52             j = 0;
53         while (j+i < len1 && str[j+i] == pat[j])
54             j++;
55         extend[i] = j;
56         id = i;
57     }
58 }
59 }
60 int main()
61 {
62     scanf("%d", &T);
63     while (T--)
64     {
65         scanf("%s%s", S, tex1);
66         int lenS = strlen(S);
67         for (int i = 0; i < lenS; i++)
68             match[(int)S[i]] = 'a'+i;
69         int len = strlen(tex1);
70         for (int i = 0; i < len; i++)
71             tex2[i] = match[(int)tex1[i]];
72         tex2[len] = 0;
73         ext_kmp(tex1, tex2);
74         for (int i = 0; i <= len; i++)
75         {
76             if ((i+extend[i] == len && i*2 >= len) || i == len)
77             {
78                 for (int j = 0; j < i; j++)
79                     printf("%c", tex1[j]);
80                 for (int j = 0; j < i; j++)
81                     printf("%c", tex2[j]);
82                 printf("\n");
83                 break;
84             }
85         }
86     }
87     return 0;
88 }

```

扩展 KMP_hdu_4333

01 /*
02 扩展 KMP 能求出一个串所有后缀串(即 $s[i \dots len]$)和模式串的最长公共前缀。于是只要将这个串复制一遍, 求出该串每个后缀与其本身的最长公共前缀即可, 当公共前缀 $\geq len$ 时, 显然相等, 否则只要比较下一位就能确定这个串与原串的大小关系。

03 至于重复串的问题, 只有当这个串有循环节的时候才会产生重复串, 用 KMP 的 next 数组求出最小循环节。
04 */
05 #include<cstdio>
06 #include<cstring>
07 #include<algorithm>
08 using namespace std;
09 const int MAXN = 100000+5, MAXM = 200000+5;
10 int T;
11 int extend[MAXN], next[MAXN], fail[MAXN];
12 char a[MAXN], aa[MAXN];
13 void get_next(char *pat)
14 {
15 next[0] = strlen(pat);
16 int k = 0;
17 while (pat[k+1] && pat[k] == pat[k+1])
18 k++;
19 next[1] = k;
20 for(int id = 1, i = 2; pat[i]; i++)
21 {
22 int u = i-id;
23 if (next[u]+i >= next[id]+id)
24 {
25 int j = next[id]+id-i;
26 if (j < 0)
27 j = 0;
28 while (pat[j+i] && pat[j] == pat[j+i])
29 j++;
30 next[i] = j;
31 id = i;
32 }
33 else
34 next[i] = next[u];
35 }
36 }
37 void ext_kmp(char *str, char *pat)
38 {
39 get_next(pat);
40 int k = 0;
41 while (str[k] && pat[k] && str[k] == pat[k])
42 k++;
43 extend[0] = k;
44 for (int id = 0, i = 1; str[i]; i++)
45 {
46 int u = i-id;
47 if (i+next[u] < extend[id]+id)
48 extend[i] = next[u];
49 else
50 {
51 int j = extend[id]+id-i;
52 if (j < 0)
53 j = 0;
54 while (str[j+i] && str[j+i] == pat[j])

```

55     j++;
56     extend[i] = j;
57     id = i;
58 }
59 }
60 }
61 void get_fail(char *pat)
62 {
63     fail[0] = -1;
64     for (int i = 1, j = -1; pat[i]; i++)
65     {
66         while (j != -1 && pat[j+1] != pat[i])
67             j = fail[j];
68         if (pat[j+1] == pat[i])
69             j++;
70         fail[i] = j;
71     }
72 }
73 int main()
74 {
75     scanf("%d", &T);
76     for (int cas = 1; cas <= T; cas++)
77     {
78         scanf("%s", a);
79         int len = strlen(a);
80         strcpy(aa, a);
81         strcpy(aa+len, a);
82         ext_kmp(aa, a);
83         get_fail(a);
84         int cir = len-fail[len-1]-1, cnt = 0;
85         //求出循环节长度 cir, 原串循环不一定完整;
86         if (len%cir)
87             cir = len;
88         for (int i = 0; i < cir; i++)
89             if (extend[i] < len && aa[i+extend[i]] < a[extend[i]])
90                 cnt++;
91         printf("Case %d: %d %d %d\n", cas, cnt, 1, cir-cnt-1);
92     }
93     return 0;
94 }

```

大数

bign-bint

```

001 //比较高效的大数
002 #include<cstdio>
003 #include<cstring>
004 using namespace std;
005 const int base = 10000; // (base^2) fit into int
006 const int width = 4; // width = log base

```

```

007 const int maxn = 1000; // n*width: 可表示的最大位数
008 struct bint
009 {
010     int len, s[maxn];
011     bint (int r = 0)
012     { // r 应该是字符串!
013         for (len = 0; r > 0; r /= base)
014             s[len++] = r%base;
015     }
016     bint &operator = (const bint &r)
017     {
018         memcpy(this, &r, (r.len+1)*sizeof(int)); // !
019         return *this;
020     }
021 };
022 bool operator < (const bint &a, const bint &b)
023 {
024     int i;
025     if (a.len != b.len) return a.len < b.len;
026     for (i = a.len-1; i >= 0 && a.s[i] == b.s[i]; i--);
027     return i < 0 ? 0 : a.s[i] < b.s[i];
028 }
029 bool operator <= (const bint &a, const bint &b)
030 {
031     return !(b < a);
032 }
033 bint operator + (const bint &a, const bint &b)
034 {
035     bint res; int i, cy = 0;
036     for (i = 0; i < a.len || i < b.len || cy > 0; i++)
037     {
038         if (i < a.len)
039             cy += a.s[i];
040         if (i < b.len)
041             cy += b.s[i];
042         res.s[i] = cy%base; cy /= base;
043     }
044     res.len = i;
045     return res;
046 }
047 bint operator - (const bint &a, const bint &b)
048 {
049     bint res; int i, cy = 0;
050     for (res.len = a.len, i = 0; i < res.len; i++)
051     {
052         res.s[i] = a.s[i]-cy;
053         if (i < b.len)
054             res.s[i] -= b.s[i];
055         if (res.s[i] < 0)
056             cy = 1, res.s[i] += base;
057         else
058             cy = 0;
059     }

```

```

060 while (res.len > 0 && res.s[res.len-1] == 0)
061     res.len--;
062 return res;
063 }
064 bint operator * (const bint &a, const bint &b)
065 {
066     bint res; res.len = 0;
067     if (0 == b.len)
068     {
069         res.s[0] = 0;
070         return res;
071     }
072     int i, j, cy;
073     for (i = 0; i < a.len; i++)
074     {
075         for (j=cy=0; j < b.len || cy > 0; j++, cy/= base)
076         {
077             if (j < b.len)
078                 cy += a.s[i]*b.s[j];
079             if (i+j < res.len)
080                 cy += res.s[i+j];
081             if (i+j >= res.len)
082                 res.s[res.len++] = cy%base;
083             else
084                 res.s[i+j] = cy%base;
085         }
086     }
087     return res;
088 }
089 bint operator / (const bint &a, const bint &b)
090 { // ! b != 0
091     bint tmp, mod, res;
092     int i, lf, rg, mid;
093     mod.s[0] = mod.len = 0;
094     for (i = a.len-1; i >= 0; i--)
095     {
096         mod = mod*base+a.s[i];
097         for (lf = 0, rg = base-1; lf < rg; )
098         {
099             mid = (lf+rg+1)/2;
100             if (b*mid <= mod)
101                 lf = mid;
102             else
103                 rg = mid-1;
104         }
105         res.s[i] = lf;
106         mod = mod-b*lf;
107     }
108     res.len = a.len;
109     while (res.len > 0 && res.s[res.len-1] == 0)
110         res.len--;
111     return res; // return mod 就是%运算
112 }

```

```

113 int digits(bint &a) // 返回位数
114 {
115     if (a.len == 0) return 0;
116     int l = (a.len-1)*4;
117     for (int t = a.s[a.len-1]; t; ++l, t/=10);
118     return l;
119 }
120 bool read(bint &b, char buf[]) // 读取失败返回 0
121 {
122     if (1 != scanf("%s", buf)) return 0;
123     int w, u, len = strlen(buf);
124     memset(&b, 0, sizeof(bint));
125     if ('0' == buf[0] && 0 == buf[1]) return 1;
126     for (w = 1, u = 0; len; )
127     {
128         u += (buf[--len]-'0')*w;
129         if (w*10 == base)
130         {
131             b.s[b.len++] = u;
132             u = 0;
133             w = 1;
134         }
135         else
136             w *= 10;
137     }
138     if (w != 1)
139         b.s[b.len++] = u;
140     return 1;
141 }
142 void write(const bint &v)
143 {
144     int i;
145     printf("%d", v.len == 0 ? 0 : v.s[v.len-1]);
146     for (i = v.len-2; i >= 0; i--)
147         printf("%04d", v.s[i]); // ! 4 == width
148     printf("\n");
149 }
150 int main()
151 {
152     freopen("input.txt", "r", stdin);
153     // freopen("output.txt", "w", stdout);
154     int a, b; scanf("%d%d", &a, &b);
155     bint A(a), B(b);
156     if (B < A)
157     {
158         write(A+B);
159         write(A-B);
160         write(A*B);
161         write(A/B);
162     }
163     return 0;
164 }

```

bign-lrj

```

001 #include<cstdio>
002 #include<iostream>
003 using namespace std;
004
005 const int maxn = 200;
006 struct bign{
007     int len, s[maxn];
008
009     bign() {
010         memset(s, 0, sizeof(s));
011         len = 1;
012     }
013
014     bign(int num) {
015         *this = num;
016     }
017
018     bign(const char* num) {
019         *this = num;
020     }
021
022     bign operator = (int num) {
023         char s[maxn];
024         sprintf(s, "%d", num);
025         *this = s;
026         return *this;
027     }
028
029     bign operator = (const char* num) {
030         len = strlen(num);
031         for(int i = 0; i < len; i++) s[i] = num[len-i-1] - '0';
032         return *this;
033     }
034
035     string str() const {
036         string res = "";
037         for(int i = 0; i < len; i++) res = (char)(s[i] + '0') + res;
038         if(res == "") res = "0";
039         return res;
040     }
041
042     bign operator + (const bign& b) const{
043         bign c;
044         c.len = 0;
045         for(int i = 0, g = 0; g || i < max(len, b.len); i++) {
046             int x = g;
047             if(i < len) x += s[i];
048             if(i < b.len) x += b.s[i];
049             c.s[c.len++] = x % 10;
050             g = x / 10;
051 }

```

```

052 return c;
053 }
054
055 void clean() {
056     while(len > 1 && !s[len-1]) len--;
057 }
058
059 bign operator * (const bign& b) {
060     bign c; c.len = len + b.len;
061     for(int i = 0; i < len; i++)
062         for(int j = 0; j < b.len; j++)
063             c.s[i+j] += s[i] * b.s[j];
064     for(int i = 0; i < c.len-1; i++){
065         c.s[i+1] += c.s[i] / 10;
066         c.s[i] %= 10;
067     }
068     c.clean();
069     return c;
070 }
071
072 bign operator - (const bign& b) {
073     bign c; c.len = 0;
074     for(int i = 0, g = 0; i < len; i++) {
075         int x = s[i] - g;
076         if(i < b.len) x -= b.s[i];
077         if(x >= 0) g = 0;
078         else {
079             g = 1;
080             x += 10;
081         }
082         c.s[c.len++] = x;
083     }
084     c.clean();
085     return c;
086 }
087
088 bool operator < (const bign& b) const{
089     if(len != b.len) return len < b.len;
090     for(int i = len-1; i >= 0; i--)
091         if(s[i] != b.s[i]) return s[i] < b.s[i];
092     return false;
093 }
094
095 bool operator > (const bign& b) const{
096     return b < *this;
097 }
098
099 bool operator <= (const bign& b) {
100     return !(b > *this);
101 }
102
103 bool operator == (const bign& b) {
104     return !(b < *this) && !(*this < b);

```

```

105     }
106
107     bign operator += (const bign& b) {
108         *this = *this + b;
109         return *this;
110     }
111 };
112
113 istream& operator >> (istream &in, bign& x) {
114     string s;
115     in >> s;
116     x = s.c_str();
117     return in;
118 }
119
120 ostream& operator << (ostream &out, const bign& x) {
121     out << x.str();
122     return out;
123 }
124
125 int main() {
126     bign a;
127     cin >> a;
128     a += "1234567891234567890000000000";
129     cout << a*2 << endl;
130     return 0;
131 }

```

bign-str

```

001 #include<cstdio>
002 #include<cstring>
003 using namespace std;
004 const int MAXSIZE = 200;
005 void Add(char *str1, char *str2, char *str3);
006 void Minus(char *str1, char *str2, char *str3);
007 void Mul(char *str1, char *str2, char *str3);
008 void Div(char *str1, char *str2, char *str3);
009 int main(void)
010 {
011     char str1[MAXSIZE], str2[MAXSIZE], str3[MAXSIZE];
012     while (scanf("%s %s", str1, str2) == 2)
013     {
014         if (strcmp(str1, "0"))
015         {
016             memset(str3, '0', sizeof(str3)); // !!!!!
017             Add(str1, str2, str3);
018             printf("%s\n", str3);
019             memset(str3, '0', sizeof(str3));
020             Minus(str1, str2, str3);
021             printf("%s\n", str3);
022             memset(str3, '0', sizeof(str3));

```

```

023             Mul(str1, str2, str3);
024             printf("%s\n", str3);
025             memset(str3, '0', sizeof(str3));
026             Div(str1, str2, str3);
027             printf("%s\n", str3);
028         }
029     else
030     {
031         if (strcmp(str2, "0"))
032             printf("%s\n-%s\n0\n0\n", str2, str2);
033         else
034             printf("0\n0\n0\n0\n");
035     }
036 }
037 return 0;
038 }
039 void Add(char *str1, char *str2, char *str3)
040 { // str3 = str1 + str2;
041     int i, j, il, i2, tmp, carry;
042     int len1 = strlen(str1), len2 = strlen(str2);
043     char ch;
044     il = len1-1; i2 = len2-1;
045     j = carry = 0;
046     for (; il >= 0 && i2 >= 0; ++j, --il, --i2)
047     {
048         tmp = str1[il]-'0'+str2[i2]-'0'+carry;
049         carry = tmp/10;
050         str3[j] = tmp%10+'0';
051     }
052     while (il >= 0)
053     {
054         tmp = str1[il--]-'0'+carry;
055         carry = tmp/10;
056         str3[j++] = tmp%10+'0';
057     }
058     while (i2 >= 0)
059     {
060         tmp = str2[i2--]-'0'+carry;
061         carry = tmp/10;
062         str3[j++] = tmp%10+'0';
063     }
064     if (carry)
065         str3[j++] = carry+'0';
066     str3[j] = '\0';
067     for (i = 0, --j; i < j; ++i, --j)
068     {
069         ch = str3[i]; str3[i] = str3[j]; str3[j] = ch;
070     }
071 }
072 void Minus(char *str1, char *str2, char *str3)
073 { // str3 = str1-str2 (str1 > str2)
074     int i, j, il, i2, tmp, carry;
075     int len1 = strlen(str1), len2 = strlen(str2);

```

```

076 char ch;
077 i1 = len1-1; i2 = len2-1;
078 j = carry = 0;
079 while (i2 >= 0)
080 {
081     tmp = str1[i1]-str2[i2]-carry;
082     if (tmp < 0)
083     {
084         str3[j] = tmp+10+'0'; carry = 1;
085     }
086     else
087     {
088         str3[j] = tmp+'0'; carry = 0;
089     }
090     --i1; --i2; ++j;
091 }
092 while (i1 >= 0)
093 {
094     tmp = str1[i1]-'0'-carry;
095     if (tmp < 0)
096     {
097         str3[j] = tmp+10+'0'; carry = 1;
098     }
099     else
100     {
101         str3[j] = tmp+'0'; carry = 0;
102     }
103     --i1; ++j;
104 }
105 --j;
106 while (str3[j] == '0' && j > 0)
107     --j;
108 str3[++j] = '\0';
109 for (i=0, --j; i < j; ++i, --j)
110 {
111     ch = str3[i]; str3[i] = str3[j]; str3[j] = ch;
112 }
113 }
114 void Mul(char *str1, char *str2, char *str3)
115 {
116     int i, j, i1, i2, tmp, carry, jj;
117     int len1 = strlen(str1), len2 = strlen(str2);
118     char ch;
119     jj = carry = 0;
120     for (i1=len1-1; i1 >= 0; --i1)
121     {
122         j = jj;
123         for (i2=len2-1; i2 >= 0; --i2, ++j)
124         {
125             tmp = (str3[j]-'0')+(str1[i1]-'0')*(str2[i2]-'0')+carry;
126             if (tmp > 9)
127             {
128                 carry = tmp/10; str3[j] = tmp%10+'0';

```

```

129     }
130     else
131     {
132         str3[j] = tmp+'0'; carry = 0;
133     }
134 }
135 if (carry)
136 {
137     str3[j] = carry+'0'; carry = 0; ++j;
138 }
139 ++jj;
140 }
141 --j;
142 while (str3[j] == '0' && j > 0)
143     --j;
144 str3[++j] = '\0';
145 for (i=0, --j; i < j; ++i, --j)
146 {
147     ch = str3[i]; str3[i] = str3[j]; str3[j] = ch;
148 }
149 }
150 void Div(char *str1, char *str2, char *str3)
151 {
152     int i1, i2, i, j, jj, tag, carry, cf, c[MAXSIZE];
153     int len1 = strlen(str1), len2 = strlen(str2), lend;
154     char d[MAXSIZE];
155     memset(c, 0, sizeof(c));
156     memcpy(d, str1, len2);
157     lend = len2; j = 0;
158     for (i1=len2-1; i1 < len1; ++i1)
159     {
160         if (lend < len2)
161         {
162             d[lend] = str1[i1+1]; c[j] = 0;
163             ++j; ++lend;
164         }
165         else if (lend == len2)
166         {
167             jj = 1;
168             for (i=0; i < lend; ++i)
169             {
170                 if (d[i] > str2[i]) break;
171                 else if (d[i] < str2[i])
172                 {
173                     jj = 0; break;
174                 }
175             }
176             if (jj == 0)
177             {
178                 d[lend] = str1[i1+1]; c[j] = 0;
179                 ++j; ++lend;
180                 continue;
181             }

```



```

182     }
183     if (jj==1 || lend > len2)
184     {
185         cf = jj=0;
186         while (d[jj] <= '0' && jj < lend)
187             ++jj;
188         if (lend-jj > len2)
189             cf = 1;
190         else if (lend-jj < len2)
191             cf = 0;
192         else
193         {
194             i2 = 0; cf = 1;
195             for (i = jj; i < lend; ++i)
196             {
197                 if (d[i] < str2[i2])
198                 {
199                     cf = 0; break;
200                 }
201                 else if (d[i] > str2[i2])
202                 {
203                     break;
204                 }
205                 ++i2;
206             }
207         } //else
208         while (cf)
209         {
210             i2 = len2-1; cf = 0;
211             for (i = lend-1; i >= lend-len2; --i)
212             {
213                 d[i] = d[i]-str2[i2]+'0';
214                 if (d[i] < '0')
215                 {
216                     d[i] = d[i]+10; carry = 1;
217                     --d[i-1];
218                 }
219                 else
220                     carry = 0;
221                 --i2;
222             }
223             ++c[j]; jj=0;
224             while (d[jj] <= '0' && jj < lend)
225                 ++jj;
226             if (lend-jj > len2)
227                 cf = 1;
228             else if (lend-jj < len2)
229                 cf = 0;
230             else
231             {
232                 i2 = 0; cf = 1;
233                 for (i = jj; i < lend; ++i)
234                 {

```

```

235                     if (d[i] < str2[i2])
236                     {
237                         cf = 0; break;
238                     }
239                     else if (d[i] > str2[i2])
240                     {
241                         break;
242                     }
243                     ++i2;
244                 }
245             } //else
246         } //while
247         jj = 0;
248         while (d[jj] <= '0' && jj < lend)
249             ++jj;
250         for (i = 0; i < lend-jj; ++i)
251             d[i] = d[i+jj];
252         d[i] = str1[i1+1]; lend = i+1;
253         ++j;
254     } //else
255 } //for
256 i = tag = 0;
257 while (c[i] == 0)
258     ++i;
259 for (; i < j; ++i, ++tag)
260     str3[tag] = c[i]+'0';
261 str3[tag] = '\0';
262 }

```

后缀数组

第K个子串_hdu_3553

```

001 #include<cstdio>
002 #include<cstring>
003 #include<algorithm>
004 #include<set>
005 using namespace std;
006 const int MAXN = 100000+5;
007 int T;
008 int sa[MAXN], height[MAXN], rank[MAXN], tmp[MAXN], top[MAXN];
009 int Tr[MAXN<2];
010 long long K, sumlen[MAXN];
011 char S[MAXN];
012 namespace SuffixArray
013 {
014     void makesa(char *s, int n)
015     {
016         int lena = n < 256 ? 256 : n;
017         memset(top, 0, lena*sizeof(int));
018         for (int i = 0; i < n; i++)

```

```

019     top[rank[i] = s[i]&(-1)]++;
020     for (int i = 1; i < lena; i++)
021         top[i] += top[i-1];
022     for (int i = 0; i < n; i++)
023         sa[--top[rank[i]]] = i;
024     for (int k = 1; k < n; k <= 1)
025     {
026         for (int i = 0; i < n; i++)
027         {
028             int j = sa[i]-k;
029             if (j < 0)
030                 j += n;
031             tmp[top[rank[j]]++] = j;
032         }
033         int j = sa[tmp[0]] = top[0] = 0;
034         for (int i = 1; i < n; i++)
035         {
036             if (rank[tmp[i]] != rank[tmp[i-1]] || rank[tmp[i]+k] !=
rank[tmp[i-1]+k])
037                 top[++j] = i;
038             sa[tmp[i]] = j;
039         }
040         memcpy(rank, sa, n*sizeof(int));
041         memcpy(sa, tmp, n*sizeof(int));
042         if (j+1 >= n)
043             break;
044     }
045 }
046 void lcp(char *s, int n)
047 {
048     height[0] = 0;
049     for (int i = 0, k = 0, j = rank[0]; i+1 < n; i++, k++)
050         while (k >= 0 && s[i] != s[sa[j-1]+k])
051         {
052             height[j] = k--;
053             j = rank[sa[j]+1];
054         }
055 }
056 }
057 namespace SegTr
058 {
059     void Build(int idx, int L, int R)
060     {
061         if (L == R)
062         {
063             Tr[idx] = R;
064             return;
065         }
066         int mid = (L+R)>>1, left = idx<<1, right = idx<<1|1;
067         Build(left, L, mid);
068         Build(right, mid+1, R);
069         Tr[idx] = (height[Tr[left]] <= height[Tr[right]] ? Tr[left]
: Tr[right]);

```

```

070     }
071     int Query(int idx, int L, int R, int l, int r)
072     {
073         if (l <= L && R <= r)
074             return Tr[idx];
075         int mid = (L+R)>>1, left = idx<<1, right = idx<<1|1;
076         int ql = 0, qr = 0;
077         if (l <= mid)
078             ql = Query(left, L, mid, l, r);
079         if (mid < r)
080             qr = Query(right, mid+1, R, l, r);
081         if (ql && !qr)
082             return ql;
083         else if (!ql && qr)
084             return qr;
085         else
086             return (height[ql] <= height[qr] ? ql : qr);
087     }
088 }
089 void solve(int len, int &rk, int &rl)
090 {
091     int h = 0;
092     long long a = 1, b = len;
093     while (a < b)
094     {
095         int q = SegTr::Query(1, 1, len, a+1, b);
096         if (K <= (height[q]-h)*(b-a+1))
097         {
098             rk = a; rl = h+1+(K-1)/(b-a+1);
099             return;
100         }
101         K -= (height[q]-h)*(b-a+1);
102         if (K <= sumlen[q-1]-sumlen[a-1]-height[q]*(q-a))
103         {
104             b = q-1; h = height[q];
105             continue;
106         }
107         K -= sumlen[q-1]-sumlen[a-1]-height[q]*(q-a);
108         a = q;
109         h = height[q];
110     }
111     rk = a; rl = h+K;
112 }
113 int main()
114 {
115     scanf("%d", &T);
116     for (int cas = 1; cas <= T; cas++)
117     {
118         scanf("%s%I64d", S, &K);
119         int len = strlen(S);
120         SuffixArray::makesa(S, len+1);
121         SuffixArray::lcp(S, len+1);
122         for (int i = 1; i <= len; i++)

```

```

123     sumlen[i] = sumlen[i-1]+len-sa[i];
124     SegTr::Build(1, 1, len);
125     int rk, rl;
126     solve(len, rk, rl);
127     printf("Case %d: ", cas);
128     for (int i = 0; i < rl; i++)
129         printf("%c", S[sa[rk]+i]);
130     printf("\n");
131 }
132 return 0;
133 }

```

多串子串并集_后缀数组_hdu_4416

```

01 /*
02 求多串的子串并集元素的个数，先用没出现过的不同的字符把多个串拼接，用后缀数组求这个串
的不同子串的个数，再减去含有拼接字符的子串的个数。用上述方法求『A、B1、.....、BN』中不同子
串 的个数 sumAB 和 『B1、.....、BN』中不同子串的个数 sumB，答案就是 sumAB-sumB。
03 */
04 #include<cstdio>
05 #include<cstring>
06 #include<algorithm>
07 using namespace std;
08 const int MAXN = 300000+5, MAXM = 100000+5;
09 int T, N, L[MAXN];
10 int len, sa[MAXN], height[MAXN], rank[MAXN], tmp[MAXN], top[MAXN];
11 int a[MAXN];
12 char A[MAXN];
13 void makesa(int *s, int n)
14 {
15     int lena = n < 256 ? 256 : n;
16     memset(top, 0, lena*sizeof(int));
17     for (int i = 0; i < n; i++)
18         top[rank[i] = s[i]&(-1)]++;
19     for (int i = 1; i < lena; i++)
20         top[i] += top[i-1];
21     for (int i = 0; i < n; i++)
22         sa[--top[rank[i]]] = i;
23     for (int k = 1; k < n; k <= 1)
24     {
25         for (int i = 0; i < n; i++)
26         {
27             int j = sa[i]-k;
28             if (j < 0)
29                 j += n;
30             tmp[top[rank[j]]++] = j;
31         }
32         int j = sa[tmp[0]] = top[0] = 0;
33         for (int i = 1; i < n; i++)
34         {
35             if (rank[tmp[i]] != rank[tmp[i-1]] || rank[tmp[i]+k] != rank[tmp[
i-1]+k])

```

```

36             top[++j] = i;
37             sa[tmp[i]] = j;
38         }
39         memcpy(rank, sa, n*sizeof(int));
40         memcpy(sa, tmp, n*sizeof(int));
41         if (j+1 >= n)
42             break;
43     }
44 }
45 void lcp(int *s, int n)
46 {
47     height[0] = 0;
48     for (int i = 0, k = 0, j = rank[0]; i+1 < n; i++, k++)
49         while (k >= 0 && s[i] != s[sa[j-1]+k])
50         {
51             height[j] = k--;
52             j = rank[sa[j]+1];
53         }
54 }
55 int main()
56 {
57     scanf("%d", &T);
58     for (int cas = 1; cas <= T; cas++)
59     {
60         scanf("%d%s", &N, A);
61         len = 0;
62         for (L[0] = 0; A[L[0]]; L[0]++)
63             a[len++] = A[L[0]]-'a'+1;
64         for (int i = 1; i <= N; i++)
65         {
66             a[len++] = 26+i;
67             scanf("%s", A);
68             for (L[i] = 0; A[L[i]]; L[i]++)
69                 a[len++] = A[L[i]]-'a'+1;
70         }
71         a[len] = 0;
72         long long sumAB = 0, sumB = 0;
73         makesa(a, len+1);
74         lcp(a, len+1);
75         for (int i = 1; i <= len; i++)
76             sumAB += len-sa[i]-height[i];
77         long long l = len;
78         for (int i = 0; i < N; i++)
79         {
80             l -= L[i];
81             sumAB -= (L[i]+1)*l;
82             l--;
83         }
84         len -= L[0]+1;
85         makesa(a+L[0]+1, len+1);
86         lcp(a+L[0]+1, len+1);
87         for (int i = 1; i <= len; i++)
88             sumB += len-sa[i]-height[i];

```

```

89     l = len;
90     for (int i = 1; i < N; i++)
91     {
92         l -= L[i];
93         sumB -= (L[i]+1)*l;
94         l--;
95     }
96     printf("Case %d: %I64d\n", cas, sumAB-sumB);
97 }
98 return 0;
99 }

```

最长重复不重叠子串_后缀数组+按 height 分组+二分_poj_1743

```

01 #include<cstdio>
02 #include<cstring>
03 #include<algorithm>
04 using namespace std;
05 const int MAXN = 20000+5;
06 const int INF = 0x3f3f3f3f;
07 int N, a[MAXN], s[MAXN];
08 int sa[MAXN], height[MAXN], rank[MAXN], tmp[MAXN], top[MAXN];
09 void makesa(int *s, int n)
10 {
11     int lena = n < 256 ? 256 : n;
12     memset(top, 0, lena*sizeof(int));
13     for (int i = 0; i < n; i++)
14         top[rank[i] = s[i]&(-1)]++;
15     for (int i = 1; i < lena; i++)
16         top[i] += top[i-1];
17     for (int i = 0; i < n; i++)
18         sa[--top[rank[i]]] = i;
19     for (int k = 1; k < n; k <= 1)
20     {
21         for (int i = 0; i < n; i++)
22         {
23             int j = sa[i]-k;
24             if (j < 0)
25                 j += n;
26             tmp[top[rank[j]]++] = j;
27         }
28         int j = sa[tmp[0]] = top[0] = 0;
29         for (int i = 1; i < n; i++)
30         {
31             if (rank[tmp[i]] != rank[tmp[i-1]] || rank[tmp[i]+k] != rank[tmp[
i-1]+k])
32                 top[++j] = i;
33             sa[tmp[i]] = j;
34         }
35         memcpy(rank, sa, n*sizeof(int));
36         memcpy(sa, tmp, n*sizeof(int));
37         if (j+1 >= n)

```

```

38             break;
39         }
40     }
41 void lcp(int *s, int n)
42 {
43     height[0] = 0;
44     for (int i = 0, k = 0, j = rank[0]; i+1 < n; i++, k++)
45         while (k >= 0 && s[i] != s[sa[j-1]+k])
46         {
47             height[j] = k--;
48             j = rank[sa[j]+1];
49         }
50 }
51 int main()
52 {
53     while (scanf("%d", &N) && N)
54     {
55         int len = 0;
56         for (int i = 0; i < N; i++)
57         {
58             scanf("%d", &a[i]);
59             if (i)
60                 s[len++] = a[i]-a[i-1]+88;
61         }
62         s[len] = 0;
63         makesa(s, len+1);
64         lcp(s, len+1);
65         int l = 4, r = max(l+1, N/2), ans = -1;
66         while (l < r)
67         {
68             int mid = (l+r)>>1, t = 0, mini = sa[0], maxi = sa[0];
69             for (int i = 1; i <= len; i++)
70             {
71                 if (height[i] >= mid)
72                 {
73                     mini = min(mini, sa[i]);
74                     maxi = max(maxi, sa[i]);
75                 }
76                 else
77                 {
78                     t = max(t, maxi-mini);
79                     mini = maxi = sa[i];
80                 }
81             }
82             t = max(t, maxi-mini);
83             if (t > mid)
84             {
85                 ans = mid;
86                 l = mid+1;
87             }
88             else
89                 r = mid;
90         }

```

```

91     printf("%d\n", ans+1);
92 }
93 return 0;
94 }

```

线段树

矩形并面积_离散化+扫描线+线段树_hdu_4419

```

01 #include<cstdio>
02 #include<cstring>
03 #include<algorithm>
04 #include<map>
05 #define left(x) x<<1
06 #define right(x) x<<1|1
07 using namespace std;
08 const int MAXN = 10000+5, MAXM = 20000+5;
09 const int ALL = 1<<3;
10 int T, N, clr[MAXN], X[MAXM], Y[MAXM], y[MAXM], r[MAXM];
11 int ID[1<<8];
12 int Tsum[ALL][MAXM<<2], Tcov[ALL][MAXM<<2];
13 char C[5];
14 bool cmp(const int a, const int b)
15 {
16     return X[a] < X[b];
17 }
18 //void Build(int idx, int L, int R)
19 //{
20 //    for (int k = 1; k < ALL; k++)
21 //        Tsum[k][idx] = Tcov[k][idx] = 0;
22 //    if (R-L == 1)
23 //        return;
24 //    int mid = (L+R)>>1;
25 //    Build(left(idx), L, mid);
26 //    Build(right(idx), mid, R);
27 //}
28 void Update(int tr, int idx, int L, int R, int l, int r, int c)
29 {
30     if (l <= L && R <= r)
31         Tcov[tr][idx] += c;
32     else
33     {
34         int mid = (L+R)>>1;
35         if (l < mid)
36             Update(tr, left(idx), L, mid, l, r, c);
37         if (mid < r)
38             Update(tr, right(idx), mid, R, l, r, c);
39     }
40     if (Tcov[tr][idx])
41         Tsum[tr][idx] = y[R-1]-y[L-1];
42     else if (R-L == 1)

```

```

43     Tsum[tr][idx] = 0;
44     else
45         Tsum[tr][idx] = Tsum[tr][left(idx)]+Tsum[tr][right(idx)];
46 }
47 int main()
48 {
49     ID['R'] = 1<<0; ID['G'] = 1<<1; ID['B'] = 1<<2;
50     scanf("%d", &T);
51     for (int cas = 1; cas <= T; cas++)
52     {
53         scanf("%d", &N);
54         for (int i = 0; i < N; i++)
55         {
56             scanf("%s%d%d%d", C, &X[left(i)], &Y[left(i)], &X[right(i)], &Y
[right(i)]);
57             clr[i] = ID[C[0]];
58             y[left(i)] = Y[left(i)];
59             y[right(i)] = Y[right(i)];
60             r[left(i)] = left(i);
61             r[right(i)] = right(i);
62         }
63         int n = N<<1;
64         sort(r, r+n, cmp);
65         sort(y, y+n);
66         map<int, int> dp;
67         for (int i = 1; i <= n; i++)
68             dp[y[i-1]] = i;
69         // Build(1, 1, n);
70         long long area[ALL] = {};
71         for (int i = 0; i < n; i++)
72             for (int k = 1; k < ALL; k++)
73             {
74                 if (k&clr[r[i]>>1])
75                     Update(k, 1, 1, n, dp[Y[left(r[i]>>1)]], dp[Y[right(r[i]>>1)]]
, (r[i]&1 ? -1 : 1));
76                 if (i+1 < n)
77                     area[k] += (long long)Tsum[k][1]*(X[r[i+1]]-X[r[i]]);
78             }
79         printf("Case %d:\n", cas);
80         printf("%I64d\n", area[7]-area[6]);
81         printf("%I64d\n", area[7]-area[5]);
82         printf("%I64d\n", area[7]-area[3]);
83         printf("%I64d\n", area[5]+area[6]-area[4]-area[7]);
84         printf("%I64d\n", area[3]+area[6]-area[2]-area[7]);
85         printf("%I64d\n", area[3]+area[5]-area[1]-area[7]);
86         printf("%I64d\n", area[1]+area[2]+area[4]-area[3]-area[5]-area[6]+a
rea[7]);
87     }
88     return 0;
89 }

```

线段树求矩形并周长_hdu_1828

```

001 /*
002 思路: 扫描线+线段树。记录完全覆盖住当前区间的线段条数, 区间左右端点被几条线段覆盖。
003 叶节点表示长度为 1 的区间。用一个查询函数求一共有多少孤立线段。
004 */
005 #include<cstdio>
006 #include<cstring>
007 #include<algorithm>
008 using namespace std;
009 const int MAXN = 20000+5, MAXM = 10000+5;
010 const int A = 10000, Len = 20000;
011 int N, x[MAXN], y[MAXN], r[MAXN];
012 int Tr[MAXN<<2], Tcov[MAXN<<2], covl[MAXN<<2], covr[MAXN<<2], mark[MA
013 XN<<2];
014 bool cmpx(const int a, const int b)
015 {
016     return x[a] < x[b];
017 }
018 bool cmpy(const int a, const int b)
019 {
020     return y[a] < y[b];
021 }
022 void Init(int idx, int L, int R)
023 {
024     if (L == R)
025     {
026         Tr[idx] = 0;
027         covl[idx] = covr[idx] = 0;
028         mark[idx] = 0;
029         return;
030     }
031     int mid = (L+R)/2;
032     Init(2*idx, L, mid);
033     Init(2*idx+1, mid, R);
034     Tr[idx] = Tr[2*idx] + Tr[2*idx+1] - (covr[2*idx] && covl[2*idx+1] ? 1 : 0);
035     covl[idx] = min(covl[2*idx], covl[2*idx+1]);
036     covr[idx] = max(covr[2*idx], covr[2*idx+1]);
037     mark[idx] = mark[2*idx] || mark[2*idx+1];
038 }
039 void PushDown(int idx, int L, int R)
040 {
041     if (L == R)
042     {
043         Tr[idx] = 0;
044         covl[idx] = covr[idx] = 0;
045         mark[idx] = 0;
046         return;
047     }
048     int mid = (L+R)/2;
049     PushDown(2*idx, L, mid);
050     PushDown(2*idx+1, mid, R);
051     Tr[idx] = Tr[2*idx] + Tr[2*idx+1] - (covr[2*idx] && covl[2*idx+1] ? 1 : 0);
052     covl[idx] = min(covl[2*idx], covl[2*idx+1]);
053     covr[idx] = max(covr[2*idx], covr[2*idx+1]);
054     mark[idx] = mark[2*idx] || mark[2*idx+1];
055 }
056 void Update(int idx, int L, int R, int l, int r, int c)
057 {
058     if (L >= l && R <= r)
059     {
060         Tr[idx] += c;
061         covl[idx] += c;
062         covr[idx] += c;
063         mark[idx] += c;
064         return;
065     }
066     int mid = (L+R)/2;
067     if (l < mid)
068         Update(2*idx, L, mid, l, r, c);
069     if (r > mid)
070         Update(2*idx+1, mid, R, l, r, c);
071     Tr[idx] = Tr[2*idx] + Tr[2*idx+1] - (covr[2*idx] && covl[2*idx+1] ? 1 : 0);
072     covl[idx] = min(covl[2*idx], covl[2*idx+1]);
073     covr[idx] = max(covr[2*idx], covr[2*idx+1]);
074     mark[idx] = mark[2*idx] || mark[2*idx+1];
075 }
076 int main()
077 {
078     while (scanf("%d", &N) != EOF)
079     {
080         for (int i = 0; i < N; i++)
081         {
082             scanf("%d%d", &x[i], &y[i]);
083             x[i] += A; y[i] += A;
084             r[i] = y[i];
085             j = 2*i+1;
086             scanf("%d%d", &x[j], &y[j]);
087             x[j] += A; y[j] += A;
088             r[j] = y[j];
089         }
090         sort(r, r+N, cmpy);
091         for (int i = 0; i < N; i++)
092         {
093             bool flag = 1;
094             for (; (flag || x[r[i]] == x[r[i-1]]) && i < N; i++)
095             {
096                 flag = 0;
097                 int k = r[i];
098                 if (!(k%2))
099                     Update(1, 0, Len, y[k], y[k^1], 1);
100                 else
101                     Update(1, 0, Len, y[k^1], y[k], -1);
102             }
103             if (i < N)
104                 Update(1, 0, Len, y[k], y[k^1], 1);
105         }
106         int ans = 0;
107         for (int i = 0; i < N; i++)
108             ans += Tr[i];
109         printf("%d\n", ans);
110     }
111 }

```

```

050 if (l <= L && R <= r)
051 {
052     Tcov[idx] += c;
053     covl[idx] += c;
054     covr[idx] += c;
055     if (Tcov[idx] || R-L == 1)
056     {
057         Tr[idx] = Tcov[idx] ? 1 : 0;
058         mark[idx] += c;
059         return;
060     }
061 }
062 if (mark[idx])
063     PushDown(idx, L, R);
064 int mid = (L+R)/2, left = 2*idx, right = 2*idx+1;
065 if (l < mid)
066     Update(left, L, mid, l, r, c);
067 if (mid < r)
068     Update(right, mid, R, l, r, c);
069 covl[idx] = covl[left];
070 covr[idx] = covr[right];
071 Tr[idx] = Tr[left]+Tr[right]-(covr[left] && covl[right] ? 1 : 0);
072 }
073 int main()
074 {
075     while (scanf("%d", &N) != EOF)
076     {
077         for (int i = 0, j; i < N; i++)
078         {
079             j = 2*i;
080             scanf("%d%d", &x[j], &y[j]);
081             x[j] += A; y[j] += A;
082             r[j] = j;
083             j = 2*i+1;
084             scanf("%d%d", &x[j], &y[j]);
085             x[j] += A; y[j] += A;
086             r[j] = j;
087         }
088         int ans = 0;
089         sort(r, r+2*N, cmpx);
090         for (int i = 0; i < 2*N; i++)
091         {
092             bool flag = 1;
093             for (; (flag || x[r[i]] == x[r[i-1]]) && i < 2*N; i++)
094             {
095                 flag = 0;
096                 int k = r[i];
097                 if (!(k%2))
098                     Update(1, 0, Len, y[k], y[k^1], 1);
099                 else
100                     Update(1, 0, Len, y[k^1], y[k], -1);
101             }
102             if (i < 2*N)

```

```

103     ans += (x[r[i]]-x[r[i-1]])*Tr[1]*2;
104 }
105 sort(r, r+2*N, cmpy);
106 for (int i = 0; i < 2*N; )
107 {
108     bool flag = 1;
109     for (; (flag || y[r[i]] == y[r[i-1]]) && i < 2*N; i++)
110     {
111         flag = 0;
112         int k = r[i];
113         if (!(k%2))
114             Update(1, 0, Len, x[k], x[k^1], 1);
115         else
116             Update(1, 0, Len, x[k^1], x[k], -1);
117     }
118     if (i < 2*N)
119         ans += (y[r[i]]-y[r[i-1]])*Tr[1]*2;
120 }
121 printf("%d\n", ans);
122 }
123 return 0;
124 }

```

线段树求体积并_hdu_3642

```

001 /*
002 题意：就是给你一些长方体，求这些长方体相交至少 3 次的体积和。
003 思路：对 z 轴扫描线，每次在 xy 平面对 x 轴扫描线、对 y 轴离散化用线段树求面积并，再把分段求得的体积加和。
004 */
005 #include<cstdio>
006 #include<cstring>
007 #include<algorithm>
008 using namespace std;
009 const int MAXN = 2000+5, MAXM = 2000+5, MAXP = 2000000+5;
010 int T, N, X[MAXN], Y[MAXN], Z[MAXN], rx[MAXN], ry[MAXN], rz[MAXN];
011 int Tr[MAXN<<2], Tcov[MAXN<<2], mark[MAXN<<2];
012 int match[MAXP], toy[MAXN];
013 bool cmpz(const int a, const int b)
014 {
015     return Z[a] < Z[b];
016 }
017 bool cmpx(const int a, const int b)
018 {
019     return X[a] < X[b];
020 }
021 bool cmpy(const int a, const int b)
022 {
023     return Y[a] < Y[b];
024 }
025 //void Init(int idx, int L, int R)
026 //{

```

```

027 // if (R-L == 1)
028 // {
029 //     Tr[idx] = 0;
030 //     Tcov[idx] = 0;
031 //     mark[idx] = 0;
032 //     return;
033 // }
034 // int mid = (L+R)/2, left = idx*2, right = idx*2+1;
035 // Init(left, L, mid);
036 // Init(right, mid, R);
037 // Tr[idx] = 0;
038 // Tcov[idx] = 0;
039 // mark[idx] = 0;
040 //}
041 void PushDown(int idx, int L, int R)
042 {
043     int mid = (L+R)/2, left = idx*2, right = idx*2+1;
044     Tcov[left] += mark[idx];
045     Tr[left] = Tcov[left] > 2 ? toy[mid]-toy[L] : 0;
046     mark[left] += mark[idx];
047     Tcov[right] += mark[idx];
048     Tr[right] = Tcov[right] > 2 ? toy[R]-toy[mid] : 0;
049     mark[right] += mark[idx];
050     mark[idx] = 0;
051 }
052 void Update(int idx, int L, int R, int l, int r, int c)
053 {
054     if (l <= L && R <= r)
055     {
056         Tcov[idx] += c;
057         if (Tcov[idx] > 2 || R-L == 1)
058         {
059             mark[idx] += c;
060             Tr[idx] = Tcov[idx] > 2 ? toy[R]-toy[L] : 0;
061             return;
062         }
063     }
064     if (mark[idx])
065         PushDown(idx, L, R);
066     int mid = (L+R)/2, left = idx*2, right = idx*2+1;
067     if (l < mid)
068         Update(left, L, mid, l, r, c);
069     if (mid < r)
070         Update(right, mid, R, l, r, c);
071     Tr[idx] = Tr[left]+Tr[right];
072 }
073 int main()
074 {
075     scanf("%d", &T);
076     for (int cas = 1; cas <= T; cas++)
077     {
078         memset(match, 0, sizeof(match));
079         scanf("%d", &N);

```

```

080 for (int i = 0; i < N; i++)
081     for (int j = 0; j < 2; j++)
082     {
083         int k = 2*i+j;
084         scanf("%d%d%d", &X[k], &Y[k], &Z[k]);
085         Y[k] += 1000000;
086         rx[k] = ry[k] = rz[k] = k;
087     }
088 sort(rx, rx+2*N, cmpx);
089 sort(ry, ry+2*N, cmpy);
090 sort(rz, rz+2*N, cmpz);
091 int cnt = 0;
092 for (int i = 0; i < 2*N; i++)
093     if (!match[Y[ry[i]]])
094     {
095         match[Y[ry[i]]] = ++cnt;
096         toy[cnt] = Y[ry[i]];
097     }
098 long long ans = 0;
099 for (int i = 0; i < 2*N; )
100 {
101     long long area = 0;
102     for (int j = 0; j < 2*N; )
103     {
104         int curX = X[rx[j]];
105         for (; curX == X[rx[j]] && j < 2*N; j++)
106         {
107             int k = rx[j]/2;
108             if (Z[2*k] <= Z[rz[i]] && Z[rz[i]] < Z[2*k+1])
109                 Update(1, 1, cnt, match[Y[2*k]], match[Y[2*k+1]], (rx[j]&1
110 ? -1 : 1));
111         }
112         if (j < 2*N)
113             area += (long long)(X[rx[j]]-X[rx[j-1]])*Tr[1];
114     }
115     int curZ = Z[rz[i]];
116     for (; curZ == Z[rz[i]] && i < 2*N; i++)
117     {
118         ans += (Z[rz[i]]-Z[rz[i-1]])*area;
119     }
120     printf("Case %d: %I64d\n", cas, ans);
121 }
122 return 0;

```

线段树区间修改单点查询_220B

01 /*
02 题意: N 个数, M 个询问, 每次问 A_i 到 A_j 里有多少个数 x 出现了 x 次。
03 思路: 离线+线段树区间修改、单点查询。按右端点将查询区间排序。扫描数列, 假设当前数 a 第 x 次出现, 那么当 $x \geq a$ 时, 区间 $[\text{pos}[a][x-a]+1, \text{pos}[a][x-a+1]]$ 上所有点+1; 当 $x > a$ 时, 区间 $[\text{pos}[a][x-a-1]+1, \text{pos}[a][x-a]]$ 上所有点-1, $\text{pos}[a][x]$ 表示数 a 第 x 次出现的位置, 为

了方便, 设所有数第一次出现的位置为 0。若当前扫描到的位置有查询区间的右端点, 则在线段树上查询左端点处的值, 即为该次查询的答案。

```

04 */
05 #include<cstdio>
06 #include<cstring>
07 #include<algorithm>
08 #include<vector>
09 using namespace std;
10 const int MAXN = 100000+5;
11 int N, M, a[MAXN], s[MAXN], t[MAXN], r[MAXN], ans[MAXN];
12 int Tr[MAXN<<2], mark[MAXN<<2];
13 vector<int> pos[MAXN];
14 bool cmp(const int a, const int b)
15 {
16     return t[a] < t[b];
17 }
18 void PushDown(int idx)
19 {
20     int left = idx<<1, right = (idx<<1)^1;
21     Tr[left] += mark[idx];
22     mark[left] += mark[idx];
23     Tr[right] += mark[idx];
24     mark[right] += mark[idx];
25     mark[idx] = 0;
26 }
27 void Update(int idx, int L, int R, int l, int r, int c)
28 {
29     if (l <= L && R <= r)
30     {
31         Tr[idx] += c;
32         mark[idx] += c;
33         return;
34     }
35     if (mark[idx])
36         PushDown(idx);
37     int mid = (L+R)>>1, left = idx<<1, right = (idx<<1)^1;
38     if (l <= mid)
39         Update(left, L, mid, l, r, c);
40     if (mid < r)
41         Update(right, mid+1, R, l, r, c);
42 }
43 int Query(int idx, int L, int R, int x)
44 {
45     if (x == L & R == x)
46         return Tr[idx];
47     if (mark[idx])
48         PushDown(idx);
49     int mid = (L+R)>>1, left = idx<<1, right = (idx<<1)^1;
50     if (x <= mid)
51         return Query(left, L, mid, x);
52     else
53         return Query(right, mid+1, R, x);
54 }

```



```

55 int main()
56 {
57     scanf("%d%d", &N, &M);
58     for (int i = 1; i <= N; i++)
59     {
60         scanf("%d", &a[i]);
61         if (a[i] <= N && !pos[a[i]].size())
62             pos[a[i]].push_back(0);
63     }
64     for (int i = 0; i < M; i++)
65     {
66         scanf("%d%d", &s[i], &t[i]);
67         r[i] = i;
68     }
69     sort(r, r+M, cmp);
70     for (int i = 1, j = 0; i <= N && j < M; i++)
71     {
72         if (a[i] <= N)
73         {
74             pos[a[i]].push_back(i);
75             if (pos[a[i]].size() > a[i])
76                 Update(1, 1, N, pos[a[i]][pos[a[i]].size()-a[i]-1]+1, pos[a[i]]
[pos[a[i]].size()-a[i]], 1);
77             if (pos[a[i]].size() > a[i]+1)
78                 Update(1, 1, N, pos[a[i]][pos[a[i]].size()-a[i]-2]+1, pos[a[i]]
[pos[a[i]].size()-a[i]-1], -1);
79         }
80         for (; t[r[j]] == i && j < M; j++)
81             ans[r[j]] = Query(1, 1, N, s[r[j]]);
82     }
83     for (int i = 0; i < M; i++)
84         printf("%d\n", ans[i]);
85     return 0;
86 }

```

最长上升子序列

二维 LIS+方案输出_sgu_521

```

01 /*
02 正向、反向分别求 LIS，再枚举每个点.....
03 */
04 #include<cstdio>
05 #include<cstring>
06 #include<algorithm>
07 #include<vector>
08 using namespace std;
09 const int MAXN = 100000+5;
10 int N, x[MAXN], y[MAXN], id[MAXN];
11 int Y[MAXN], f[MAXN], d[2][MAXN], cnt[MAXN];
12 bool mark[MAXN];

```

```

13 bool cmp (const int &a, const int &b)
14 {
15     if (x[a] != x[b])
16         return x[a] < x[b];
17     else
18         return y[a] > y[b];
19 }
20 int LIS(int x)
21 {
22     int maxi = 0;
23     for (int i = 1; i <= N; i++)
24     {
25         int j = lower_bound(f+1, f+1+maxi, Y[i])-f;
26         maxi = max(maxi, j);
27         f[j] = Y[i];
28         d[x][i] = j;
29     }
30     return maxi;
31 }
32 int main()
33 {
34     while (scanf("%d", &N) != EOF)
35     {
36         memset(cnt, 0, sizeof(cnt));
37         for (int i = 1; i <= N; i++)
38         {
39             scanf("%d%d", &x[i], &y[i]);
40             id[i] = i;
41         }
42         sort(id+1, id+1+N, cmp);
43         for (int i = 1; i <= N; i++)
44             Y[i] = y[id[i]];
45         int maxlen = LIS(0);
46         for (int i = 1; i <= N; i++)
47             Y[i] = -y[id[N-i+1]];
48         LIS(1);
49         vector<int> ans[2];
50         for (int i = 1; i <= N; i++)
51         {
52             mark[i] = (d[0][i]+d[1][N-i+1] == maxlen+1);
53             if (mark[i])
54             {
55                 cnt[d[0][i]]++;
56                 ans[0].push_back(id[i]);
57             }
58         }
59         for (int i = 1; i <= N; i++)
60             if (mark[i] && cnt[d[0][i]] == 1)
61                 ans[1].push_back(id[i]);
62         for (int i = 0; i < 2; i++)
63         {
64             sort(ans[i].begin(), ans[i].end());
65             printf("%u", ans[i].size());

```

```

66     for (vector<int>::iterator it = ans[i].begin(); it != ans[i].end(
); it++)
67         printf(" %d", *it);
68     printf("\n");
69 }
70 }
71 return 0;
72 }

```

某矩形的 LIS_bupt_394

```

01 /*
02 离线读入所有点（左下、右上），在左下点查询，右上点更新。
03 */
04 #include<cstdio>
05 #include<cstring>
06 #include<algorithm>
07 using namespace std;
08 const int MAXN = 100000+5, MAXM = 200000+5;
09 int T, N, x[MAXN], y[MAXN], id[MAXN];
10 int f[MAXN], g[MAXN];
11 bool cmp (const int &a, const int &b)
12 {
13     if (x[a] != x[b])
14         return x[a] < x[b];
15     else
16         return y[a] > y[b];
17 }
18 int LIS(int n)
19 {
20     int maxi = 0;
21     for (int i = 0; i < n; i++)
22     {
23         if (!(id[i]&1))
24             g[id[i]>>1] = lower_bound(f+1, f+1+maxi, y[id[i]])-f;
25         else
26         {
27             if (g[id[i]>>1] > maxi)
28                 f[++maxi] = y[id[i]];
29             else
30                 f[g[id[i]>>1]] = min(f[g[id[i]>>1]], y[id[i]]);
31         }
32     }
33     return maxi;
34 }
35 int main()
36 {
37     scanf("%d", &T);
38     while (T--)
39     {
40         scanf("%d", &N);
41         for (int i = 0; i < N; i++)

```

```

42     {
43         scanf("%d%d%d%d", &x[i<<1], &y[i<<1], &x[i<<1|1], &y[i<<1|1]);
44         id[i<<1] = i<<1;
45         id[i<<1|1] = i<<1|1;
46     }
47     int n = N<<1;
48     sort(id, id+n, cmp);
49     printf("%d\n", LIS(n));
50 }
51 return 0;
52 }

```

最长上升子序列_poj_3903

```

01 #include<cstdio>
02 #include<cstring>
03 #include<algorithm>
04 using namespace std;
05 const int MAXN = 100000+5;
06 const int INF = 0x7fffffff;
07 int N, a[MAXN], f[MAXN];
08 //int d[MAXN];
09 int main()
10 {
11     while (scanf("%d", &N) != EOF)
12     {
13         int maxi = 0;
14         for (int i = 1; i <= N; i++)
15         {
16             scanf("%d", &a[i]);
17             int x = lower_bound(f+1, f+1+maxi, a[i])-f;
18             maxi = max(maxi, x);
19             f[x] = a[i];
20             // d[i] = x;
21         }
22         printf("%d\n", maxi);
23     }
24     return 0;
25 }

```

Mahjong_hdu_4431

```

001 #include<cstdio>
002 #include<cstring>
003 #include<algorithm>
004 #include<vector>
005 using namespace std;
006 const int MAX = 34;
007 const char *mahjong[] = {
008     "1m", "2m", "3m", "4m", "5m", "6m", "7m", "8m", "9m",
009     "1s", "2s", "3s", "4s", "5s", "6s", "7s", "8s", "9s",

```

```

010  "1p", "2p", "3p", "4p", "5p", "6p", "7p", "8p", "9p",
011  "1c", "2c", "3c", "4c", "5c", "6c", "7c"
012 };
013 int T, cnt[MAX];
014 char tile[10];
015 int id(char *s)
016 {
017     if (s[1] == 'm')
018         return s[0] - '1';
019     else if (s[1] == 's')
020         return 9 + s[0] - '1';
021     else if (s[1] == 'p')
022         return 18 + s[0] - '1';
023     else
024         return 27 + s[0] - '1';
025 }
026 //bool check_standard_dfs(int dep)
027 //{
028 //    if (dep == 5)
029 //        return 1;
030 //    bool res = 0;
031 //    if (!dep)
032 //    {
033 //        for (int i = 0; i < MAX && !res; i++) if (cnt[i] >= 2)
034 //        {
035 //            cnt[i] -= 2;
036 //            res = check_standard_dfs(dep+1);
037 //            cnt[i] += 2;
038 //        }
039 //    }
040 //    else
041 //    {
042 //        for (int i = 0; i < MAX && !res; i++)
043 //        {
044 //            if (cnt[i] >= 3)
045 //            {
046 //                cnt[i] -= 3;
047 //                res = check_standard_dfs(dep+1);
048 //                cnt[i] += 3;
049 //            }
050 //            if (i < 27 && i%9 <= 6 && cnt[i] >= 1 && cnt[i+1] >= 1 && cnt[i+2] >= 1)
051 //            {
052 //                for (int j = 0; j < 3; j++)
053 //                    cnt[i+j]--;
054 //                res = check_standard_dfs(dep+1);
055 //                for (int j = 0; j < 3; j++)
056 //                    cnt[i+j]++;
057 //            }
058 //        }
059 //    }
060 //    return res;
061 //}

```

```

062 bool check_standard()
063 {
064     bool res = 0;
065     for (int i = 0; i < MAX && !res; i++) if (cnt[i] >= 2)
066     {
067         int tmp[MAX], num = 0;
068         memcpy(tmp, cnt, sizeof(cnt));
069         tmp[i] -= 2;
070         for (int j = 0; j < MAX; j++)
071         {
072             if (tmp[j] >= 3)
073             {
074                 tmp[j] -= 3;
075                 num++;
076             }
077             if (j < 27 && j%9 < 7)
078             {
079                 while (tmp[j] >= 1 && tmp[j+1] >= 1 && tmp[j+2] >= 1)
080                 {
081                     for (int k = 0; k < 3; k++)
082                         tmp[j+k]--;
083                     num++;
084                 }
085             }
086         }
087         res = (num == 4);
088     }
089     return res;
090 }
091 bool check_ChiiToitsu()
092 {
093     for (int i = 0; i < MAX; i++)
094         if (cnt[i] && cnt[i] != 2)
095             return 0;
096     return 1;
097 }
098 bool check_KokushiMuso()
099 {
100     int res = 0;
101     for (int i = 0; i < 3; i++)
102     {
103         if (cnt[i*9+0] >= 1 && cnt[i*9+8] >= 1)
104             res += cnt[i*9+0] + cnt[i*9+8];
105         else
106             return 0;
107     }
108     for (int i = 27; i < MAX; i++)
109     {
110         if (cnt[i] >= 1)
111             res += cnt[i];
112         else
113             return 0;
114     }

```

```

115 return (res == 14);
116 }
117 int main()
118 {
119     scanf("%d", &T);
120     while (T--)
121     {
122         memset(cnt, 0, sizeof(cnt));
123         for (int i = 0; i < 13; i++)
124         {
125             scanf("%s", tile);
126             cnt[id(tile)]++;
127         }
128         vector<int> ans;
129         for (int i = 0; i < MAX; i++) if (cnt[i] < 4)
130         {
131             cnt[i]++;
132             if (check_KokushiMuso() || check_ChiiToitsu()
133             || check_standard())
134                 ans.push_back(i);
135             cnt[i]--;
136         }
137         if (ans.size())
138         {
139             printf("%d", (int)ans.size());
140             for (int i = 0; i < (int)ans.size(); i++)
141                 printf(" %s", mahjong[ans[i]]);
142             printf("\n");
143         }
144         else
145             printf("Nooten\n");
146     }
147     return 0;
148 }

```

RMQ-ST

```

01 #include<cstdio>
02 #include<cstring>
03 #include<cmath>
04 #include<algorithm>
05 using namespace std;
06 const int MAXN = 50000+5, MAXM = 16;
07 int N, Q;
08 int a[MAXN], st[MAXN][MAXM];
09 int pow2[MAXM];
10 inline int Most(const int &a, const int &b)
11 {
12     return a > b ? a : b;
13 }
14 void InitRMQ(const int &n)
15 {

```

```

16     pow2[0] = 1;
17     for (int i = 1; i <= MAXM; i++)
18         pow2[i] = pow2[i-1]<<1; //预处理 2 的 i 次方, 最大次幂要大于 MAXN
19     for (int i = 1; i <= n; i++)
20         stmax[i][0] = a[i];
21     int k = int(log(double(n))/log(2.0))+1;
22     for (int j = 1; j < k; j++)
23         for (int i = 1; i <= n; i++)
24         {
25             if (i+pow2[j-1]-1 <= n)
26                 stmax[i][j] = Most(stmax[i][j-1], stmax[i+pow2[j-1]][j-1]);
27             else
28                 break; // st[i][j] = st[i][j-1];
29         }
30 }
31 int Query(int x, int y) // x, y 均为下标:1...n
32 {
33     int k = int(log(double(y-x+1))/log(2.0));
34     return Most(stmax[x][k], stmax[y-pow2[k]+1][k]);
35 }
36 int main()
37 {
38     scanf("%d%d", &N, &Q);
39     for (int i = 1; i <= N; i++)
40         scanf("%d", &a[i]);
41     InitRMQ(N);
42     while (Q--)
43     {
44         int A, B;
45         scanf("%d%d", &A, &B);
46         int ans = Query(A, B);
47     }
48     return 0;
49 }

```

Trie 树_编辑距离阈值匹配_UVALive_4769

```

01 /*
02 求字典中存在前缀与查询串编辑距离小于阈值的词的个数
03 */
04 #include<cstdio>
05 #include<cstring>
06 #include<algorithm>
07 #include <iostream>
08 using namespace std;
09 const int MAXM = 10+5;
10 const int MAX_NODE = 3000000+5, MAX_CHD = 26;
11 int N, M, edth;
12 int nv, chd[MAX_NODE][MAX_CHD], out[MAX_NODE], ID[1<<8];
13 int vis[MAX_NODE], mark[MAX_NODE];
14 char word[MAXM];
15 namespace Trie

```

```

16 {
17 void Initialize()
18 {
19     for (int k = 0; k < MAX_CHD; k++)
20         ID[k+'a'] = k;
21 }
22 void Reset()
23 {
24     memset(chd[0], 0, sizeof(chd[0]));
25     nv = 1;
26 }
27 void Insert(char *pat)
28 {
29     int u = 0;
30     for (int i = 0; pat[i]; i++)
31     {
32         int c = ID[pat[i]];
33         if (!chd[u][c])
34         {
35             memset(chd[nv], 0, sizeof(chd[nv]));
36             out[nv] = 0;
37             chd[u][c] = nv++;
38         }
39         u = chd[u][c];
40         out[u]++;
41     }
42 }
43 }
44 void dfs(int u, char *p, int d, int c)
45 {
46     vis[u] = c;
47     if (!(*p))
48         mark[u] = c;
49     if (mark[u] == c)
50         return;
51     if (chd[u][ID[*p]])
52         dfs(chd[u][ID[*p]], p+1, d, c);
53     if (d)
54     {
55         for (int i = 0; i < MAX_CHD; i++) if (chd[u][i])
56             dfs(chd[u][i], p, d-1, c);
57         for (int i = 0; i < MAX_CHD; i++) if (chd[u][i])
58             dfs(chd[u][i], p+1, d-1, c);
59         dfs(u, p+1, d-1, c);
60     }
61 }
62 int calc(int u, int c)
63 {
64     if (vis[u] != c)
65         return 0;
66     if (mark[u] == c)
67         return out[u];
68     int res = 0;

```

```

69     for (int i = 0; i < MAX_CHD; i++) if (chd[u][i])
70         res += calc(chd[u][i], c);
71     return res;
72 }
73 int main()
74 {
75     scanf("%d", &N);
76     Trie::Initialize();
77     Trie::Reset();
78     for (int i = 1; i <= N; i++)
79     {
80         scanf("%s", word);
81         Trie::Insert(word);
82     }
83     scanf("%d", &M);
84     for (int i = 1; i <= M; i++)
85     {
86         scanf("%s%d", word, &edth);
87         dfs(0, word, edth, i);
88         printf("%d\n", calc(0, i));
89     }
90     return 0;
91 }

```

编辑距离+BK树_hdu_4323

```

01 /*
02 1.dp 求编辑距离
03 2.bk 树找相差 d 的单词
04 */
05 #include<cstdio>
06 #include<cstring>
07 #include<iostream>
08 #include<algorithm>
09 #include<queue>
10 using namespace std;
11 const int MAXN = 1500+5, MAXM = 10+5, MAXP = 400+5;
12 const int INF = 0x3f3f3f3f;
13 int T, n, m, t, cnt;
14 int d[MAXN][MAXM], next[MAXN][MAXM];
15 char str1[MAXN][MAXM], str2[MAXM];
16 int Distance(char *s1, char *s2)
17 {
18     int l1 = strlen(s1), l2 = strlen(s2);
19     for (int i = 0; i <= l1; i++)
20         for (int j = 0; j <= l2; j++)
21         {
22             if (!(i*j))
23                 d[i][j] = i+j;
24             else
25             {
26                 d[i][j] = min(d[i-1][j]+1, d[i][j-1]+1);

```

```

27     if (s1[i-1] == s2[j-1])
28         d[i][j] = min(d[i][j], d[i-1][j-1]);
29     else
30         d[i][j] = min(d[i][j], d[i-1][j-1]+1);
31     }
32     // printf("%d,%d:%d\n", i, j, d[i][j]);
33 }
34 return d[l1][l2];
35 }
36 void dfs(int u)
37 {
38     int dis = Distance(str1[u], str2);
39     if (u && dis <= t)
40         cnt++;
41     for (int k = dis-t; k <= dis+t; k++)
42         if (k >= 0 && next[u][k])
43             dfs(next[u][k]);
44 }
45 int main()
46 {
47     scanf("%d", &T);
48     for (int cas = 1; cas <= T; cas++)
49     {
50         memset(next, 0, sizeof(next));
51         scanf("%d%d", &n, &m);
52         strcpy(str1[0], "");
53         for (int i = 1; i <= n; i++)
54         {
55             scanf("%s", str1[i]);
56             for (int j = 0; ; )
57             {
58                 int dis = Distance(str1[i], str1[j]);
59                 if (!next[j][dis])
60                 {
61                     next[j][dis] = i;
62                     break;
63                 }
64                 j = next[j][dis];
65             }
66         }
67         printf("Case #%d:\n", cas);
68         for (int i = 1; i <= m; i++)
69         {
70             scanf("%s%d", str2, &t);
71             cnt = 0;
72             dfs(0);
73             printf("%d\n", cnt);
74         }
75     }
76     return 0;
77 }

```

后缀自动机_SPOJ_LCS2

```

001 #include<cstdio>
002 #include<cstring>
003 #include<algorithm>
004 using namespace std;
005 const int MAXN = 100000+5, MAXM = 10+5;
006 char s[MAXN];
007
008 //MAX_NODE = StringLength*2
009 const int MAX_NODE = 500000+5;
010 //字符集大小,一般字符形式的题 26 个
011 const int MAX_CHD = 26;
012 //已使用节点个数
013 int nv;
014 //每个节点的儿子,即当前节点的状态转移
015 int chd[MAX_NODE][MAX_CHD];
016 //此节点代表最长串的长度
017 int ml[MAX_NODE];
018 //父亲/失败指针
019 int fa[MAX_NODE];
020 //字母对应的 id
021 int id[1<<8];
022
023 //特定题目需要
024 int mml[MAX_NODE][MAXM], r[MAX_NODE];
025
026 namespace Suffix_Automaton
027 {
028     //初始化,计算字母对应的儿子 id,如:'a'->0 ... 'z'->25
029     void Initialize()
030     {
031         for (int i = 0; i < MAX_CHD; i++)
032             id['a'+i] = i;
033     }
034     //增加一个节点
035     void Add(int u, int _ml, int _fa, int v = -1)
036     {
037         ml[u] = _ml; fa[u] = _fa;
038         if (v == -1)
039             memset(chd[u], -1, sizeof(chd[u]));
040         else
041             memcpy(chd[u], chd[v], sizeof(chd[v]));
042     }
043     //建立后缀自动机
044     void Construct(char *str)
045     {
046         nv = 1; Add(0, 0, -1);
047         int cur = 0;
048         for (int i = 0; str[i]; i++)
049         {
050             int c = id[str[i]], p = cur;
051             cur = nv++; Add(cur, i+1, -1);

```

```

052     for (; p != -1 && chd[p][c] == -1; p = fa[p])
053         chd[p][c] = cur;
054     if (p == -1)
055         fa[cur] = 0;
056     else
057     {
058         int q = chd[p][c];
059         if (ml[q] == ml[p]+1)
060             fa[cur] = q;
061         else
062         {
063             int r = nv++; Add(r, ml[q], fa[q], q);
064             ml[r] = ml[p]+1; fa[q] = fa[cur] = r;
065             for (; p != -1 && chd[p][c] == q; p = fa[p])
066                 chd[p][c] = r;
067         }
068     }
069 }
070 }
071 }
072
073 bool cmp(const int &a, const int &b)
074 {
075     return ml[a] > ml[b];
076 }
077 int main()
078 {
079     Suffix_Automaton::Initialize();
080     scanf("%s", s);
081     Suffix_Automaton::Construct(s);
082     for (int i = 0; i < nv; i++)
083         r[i] = i;
084     sort(r, r+nv, cmp);
085     memset(mml, 0, sizeof(mml));
086     int cnt = 0;
087     for (int i = 1; scanf("%s", s) != EOF; i++, cnt++)
088     {
089         int l = 0, u = 0;
090         for (int j = 0; s[j]; j++)
091         {
092             int c = id[s[j]];
093             if (chd[u][c] != -1)
094                 l++, u = chd[u][c];
095             else
096             {
097                 while (u != -1 && chd[u][c] == -1)
098                     u = fa[u];
099                 if (u != -1)
100                     l = ml[u]+1, u = chd[u][c];
101                 else
102                     l = 0, u = 0;
103             }
104             mml[u][i] = max(mml[u][i], l);

```

```

105     }
106 }
107 int ans = 0;
108 for (int i = 0; i < nv; i++)
109 {
110     int mini = ml[r[i]];
111     for (int j = 1; j <= cnt; j++)
112     {
113         mini = min(mini, mml[r[i]][j]);
114         mml[fa[r[i]]][j] = max(mml[fa[r[i]]][j], mml[r[i]][j]);
115     }
116     ans = max(ans, mini);
117 }
118 printf("%d\n", ans);
119 return 0;
120 }

```

斯坦纳树_hdu_4085

```

001 /*
002 斯坦纳树
003 最后的答案可能是一个森林，所以我们要先求出斯坦纳树后进行 DP。转移的时候要注意一点，
004 只有人的个数和房子的个数相等的时候才算合法状态，所以我们要加一个 check() 函数进行检查。
005 */
006 #include<cstdio>
007 #include<cstring>
008 #include<algorithm>
009 #include<queue>
010 using namespace std;
011 const int MAXN = 50+5, MAXM = 2000+5;
012 const int INF = 0x3f3f3f3f;
013 int T, N, M, K, X, Y, Z;
014 int bit[MAXN], head[MAXN], e, next[MAXM], v[MAXM], w[MAXM];
015 int inq[MAXN][1<<MAX], d[MAXN][1<<MAX], dp[1<<MAX];
016 queue<int> Q;
017 void addedge(int x, int y, int z)
018 {
019     v[e] = y; w[e] = z;
020     next[e] = head[x]; head[x] = e++;
021 }
022 void init()
023 {
024     e = 0;
025     memset(head, -1, sizeof(head));
026     memset(d, 0x3f, sizeof(d));
027     memset(bit, 0, sizeof(bit));
028     memset(inq, 0, sizeof(inq));
029     memset(dp, 0x3f, sizeof(dp));
030 }
031 void spfa()
032 {

```

```

033 while (!Q.empty())
034 {
035     int u = Q.front() & ((1 << MAX) - 1), st = Q.front() >> MAX;
036     Q.pop();
037     inq[u][st] = 0;
038     for (int i = head[u]; i != -1; i = next[i])
039     {
040         int nst = st | bit[v[i]];
041         if (d[u][st] + w[i] < d[v[i]][nst])
042         {
043             d[v[i]][nst] = d[u][st] + w[i];
044             if (nst == st && !inq[v[i]][nst])
045             {
046                 Q.push(nst << MAX | v[i]);
047                 inq[v[i]][nst] = 1;
048             }
049         }
050     }
051 }
052 }
053 bool check(int st)
054 {
055     int res = 0;
056     for (int i = 0; i < K; i++)
057     {
058         if (st & (1 << i))
059             res++;
060         if (st & (1 << (K + i)))
061             res--;
062     }
063     return !res;
064 }
065 int main()
066 {
067     freopen("put.in", "r", stdin);
068     scanf("%d", &T);
069     while (T--)
070     {
071         init();
072         scanf("%d%d%d", &N, &M, &K);
073         for (int i = 0; i < M; i++)
074         {
075             scanf("%d%d%d", &X, &Y, &Z);
076             addedge(X, Y, Z);
077             addedge(Y, X, Z);
078         }
079         int tot = (1 << (K << 1)) - 1;
080         for (int i = 1; i <= K; i++)
081         {
082             bit[i] = 1 << (i - 1);
083             d[i][bit[i]] = 0;
084             bit[N - K + i] = 1 << (K + i - 1);
085             d[N - K + i][bit[N - K + i]] = 0;

```

```

086     }
087     for (int i = 0; i <= tot; i++)
088     {
089         for (int j = 1; j <= N; j++)
090         {
091             for (int k = (i - 1) & i; k; k = (k - 1) & i) //枚举 i 的所有子集
092                 d[j][i] = min(d[j][i], d[j][k | bit[j]] + d[j][(i - k) | bit[j]]);
093             if (d[j][i] < INF)
094             {
095                 Q.push(i << MAX | j);
096                 inq[j][i] = 1;
097             }
098         }
099         spfa();
100     }
101     for (int i = 0; i <= tot; i++)
102         for (int j = 1; j <= N; j++)
103             dp[i] = min(dp[i], d[j][i]);
104     for (int i = 0; i <= tot; i++) if (check(i))
105         for (int j = (i - 1) & i; j; j = (j - 1) & i) if (check(j))
106             dp[i] = min(dp[i], dp[j] + dp[i - j]);
107     if (dp[tot] < INF)
108         printf("%d\n", dp[tot]);
109     else
110         printf("No solution\n");
111 }
112 return 0;
113 }

```

最大非空连续和+方案_hdu_1003

```

01 #include<stdio>
02 #include<string>
03 #include<algorithm>
04 using namespace std;
05 const int MAXN = 100000 + 5;
06 const int INF = 0x3f3f3f3f;
07 int T, N, a, s, t;
08 int main()
09 {
10     scanf("%d", &T);
11     for (int cas = 1; cas <= T; cas++)
12     {
13         scanf("%d", &N);
14         int sum = 0, mini = 0, maxi = -INF, p = 1;
15         for (int i = 1; i <= N; i++)
16         {
17             scanf("%d", &a);
18             sum += a;
19             if (sum - mini > maxi)
20             {
21                 maxi = sum - mini;

```



```
22     s = p;
23     t = i;
24 }
25 if (sum < mini)
26 {
27     mini = sum;
28     p = i+1;
29 }
30 }
31 if (cas > 1)
32     printf("\n");
33 printf("Case %d:\n", cas);
34 printf("%d %d %d\n", maxi, s, t);
35 }
36 return 0;
37 }
```