 //首先声明这并不是本人原创，只不不过本人想总结出来方便大家，全为C++代码。

1.结点选择

问题描述

有一棵 n 个节点的树，树上每个节点都有一个正整数权值。如果一个点被选择了，那么在树上和它相邻的点都不能被选择。求选出的点的权值和最大是多少？

解题思路：

这题模型是树形动态规划入门题目，

dp[i][0]表示该节点不被选择，dp[i][1]表示该结点被选择。

转移方程为:

dp[u][1]+=dp[v][0];//选择了u结点，则与它邻接的结点不选；

dp[u][0]+=max(dp[v][0]，dp[v][1]);不选择u结点，则与它邻接的结点选择结果最大的；

应该特别注意：该题结点数量较大，应该选用邻接表存储边的关系

1. #include<cstdio>
2. #include<cstring>
3. #define max(a,b) ((a)>(b)?(a):(b))
4. #define maxn 100010
5. **bool** vis[maxn];
6. **int** dp[maxn][2];
7. **int** father[maxn];
8. **int** head[maxn];
9. **int** n;
10. **int** cnt;
11. **struct** Edge
12. {
13. **int** to,next;
14. }edge[2\*maxn];
15. **void** add(**int** u,**int** v)
16. {
17. edge[cnt].to=v;
18. edge[cnt].next=head[u];
19. head[u]=cnt++;
20. }
21. **void** treedp(**int** u)
22. {
23. vis[u]=1;
24. **for**(**int** i=head[u];i!=-1;i=edge[i].next)
25. {
26. **int** v=edge[i].to;
27. **if**(!vis[v])
28. {
29. treedp(v);
30. dp[u][1]+=dp[v][0];
31. dp[u][0]+=max(dp[v][1],dp[v][0]);
32. }
33. }
34. }
35. **void** init()
36. {
37. cnt=0;
38. memset(dp,0,**sizeof**(dp));
39. memset(father,0,**sizeof**(father));
40. memset(vis,0,**sizeof**(vis));
41. memset(head,-1,**sizeof**(head));
42. }
43. **int** main()
44. {
45. init();
46. scanf("%d",&n);
47. **for**(**int** i=1;i<=n;i++)
48. scanf("%d",&dp[i][1]);
49. **int** root=0;
50. **int** begin=1;
51. **for**(**int** i=0;i<n-1;i++)
52. {
53. **int** a,b;
54. scanf("%d%d",&a,&b);
55. add(a,b);
56. add(b,a);
57. father[b]=a;
58. **if**(root==b||begin)
59. {
60. root=a;
61. }
62. }
64. **while**(father[root])
65. root=father[root];
66. treedp(root);
67. **int** ans;
68. ans=max(dp[root][0],dp[root][1]);
69. printf("%d\n",ans);
70. }

2.K好数

  问题描述

如果一个自然数N的K进制表示中任意的相邻的两位都不是相邻的数字，那么我们就说这个数是K好数。求L位K进制数中K好数的数目。例如K = 4，L = 2的时候，所有K好数为11、13、20、22、30、31、33 共7个。由于这个数目很大，请你输出它对1000000007取模后的值。

解题思路：

dp[i][j]表示第i位为j的时候的个I好数个数；

因此有转移方程：

dp[i][j]=dp[i-1][m]+dp[i][j];m为上一位的值，满足的条件应为m-j的绝对值不为1.即不相邻；

应当注意的是：最后在求和的时候不能简单的统计dp[l][m] 0<=m<k;因为首位如果是0的话，其实不足L位了，所以0<m<k,也许有人会疑问这是不统计L位的0，不是第一位呀。。。。。。其实仔细想想，是等效的。

1. #include<cstdio>
2. #include<cstring>
3. #define mod 1000000007
4. #define abs(a,b) ((a)>(b)?(a-b):(b-a))
5. **int** dp[110][110];
6. **int** main()
7. {
8. **int** k,l;
9. memset(dp,0,**sizeof**(dp));
10. scanf("%d%d",&k,&l);
11. **for**(**int** i=0;i<k;i++)
12. dp[1][i]=1;
13. **for**(**int** i=2;i<=l;i++)
14. {
15. **for**(**int** j=0;j<k;j++)
16. {
17. **for**(**int** m=0;m<k;m++)
18. {
19. **if**(abs(m,j)!=1)
20. dp[i][j]=(dp[i][j]%mod+dp[i-1][m]%mod)%mod;
21. }
22. }
23. }
24. **int** ans=0;
25. **for**(**int** i=1;i<k;i++)
26. ans=(ans%mod+dp[l][i]%mod)%mod;
27. printf("%d\n",ans);
28. }

3. [操作格子](http://blog.csdn.net/buctyyzyn/article/details/43490983)

问题描述

有n个格子，从左到右放成一排，编号为1-n。

共有m次操作，有3种操作类型：

1.修改一个格子的权值，

2.求连续一段格子权值和，

3.求连续一段格子的最大值。

对于每个2、3操作输出你所求出的结果。

解题思路：

这是线段树的入门题目。直接建立线段树，求解就OK了

1. #include<cstdio>
2. #define max(a,b) ((a)>(b)?(a):(b))
3. #define min(a,b) ((a)<(b)?(a):(b))
4. #define N 100010
5. #define L(a) (a)<<1
6. #define R(a) (a)<<1|1
7. **struct** node
8. {
9. **int** l,r,maxn,minn,sum;
10. }line[3\*N];
11. **int** num[N];
12. **int** first,second;
13. **void** create(**int** k,**int** x,**int** y)
14. {
15. line[k].l=x;line[k].r=y;
16. **if**(x==y)
17. {
18. line[k].sum=num[x];
19. line[k].maxn=line[k].minn=num[x];
20. **return**;
21. }
22. **int** mid=(x+y)>>1;
23. create(L(k),x,mid);
24. create(R(k),mid+1,y);
25. line[k].maxn=max(line[L(k)].maxn,line[R(k)].maxn);
26. //line[k].minn=mymin(line[L(k)].minn,line[R(k)].minn);
27. line[k].sum=line[L(k)].sum+line[R(k)].sum;
28. }
29. **void** updata(**int** k,**int** a,**int** b)
30. {
31. **if**(line[k].l==line[k].r)
32. {
33. line[k].maxn=b;
34. line[k].sum=b;
35. **return** ;
36. }
37. **int** mid=(line[k].l+line[k].r)>>1;
38. **if**(a<=mid)
39. updata(L(k),a,b);
40. **else**
41. updata(R(k),a,b);
42. line[k].sum=line[L(k)].sum+line[R(k)].sum;
43. line[k].maxn=max(line[L(k)].maxn,line[R(k)].maxn);
44. }
45. **void** quary(**int** k,**int** x,**int** y)
46. {
47. **if**(line[k].l==x&&line[k].r==y)
48. {
49. second+=line[k].sum;
50. first=max(first,line[k].maxn);
51. **return**;
52. }
53. **int** mid=(line[k].l+line[k].r)>>1;
54. **if**(y<=mid)
55. quary(L(k),x,y);
56. **else** **if**(x>mid)
57. quary(R(k),x,y);
58. **else**
59. {
60. quary(L(k),x,mid);
61. quary(R(k),mid+1,y);
62. }
63. }
64. **int** main()
65. {
66. **int** n,m;
67. scanf("%d%d",&n,&m);
68. **for**(**int** i=1;i<=n;i++)
69. scanf("%d",&num[i]);
70. create(1,1,n);
71. **for**(**int** i=1;i<=m;i++)
72. {
73. **int** a,b,c;
74. scanf("%d%d%d",&a,&b,&c);
75. **if**(a==1)
76. {
77. updata(1,b,c);
78. }
79. **else** **if**(a==2)
80. {
81. first=-10000;
82. second=0;
83. quary(1,b,c);
84. printf("%d\n",second);
85. }
86. **else**
87. {
88. first=-10000;
89. second=0;
90. quary(1,b,c);
91. printf("%d\n",first);
92. }
94. }
95. }

4. [2n皇后](http://blog.csdn.net/buctyyzyn/article/details/43491049)

问题描述

　　给定一个n\*n的棋盘，棋盘中有一些位置不能放皇后。现在要向棋盘中放入n个黑皇后和n个白皇后，使任意的两个黑皇后都不在同一行、同一列或同一条对角线上，任意的两个白皇后都不在同一行、同一列或同一条对角线上。问总共有多少种放法？n小于等于8。

   解题思路：

   深搜黑皇后的可能分布，达到n时候深搜白皇后。

1. #include<cstdio>
2. #include<iostream>
3. #include<cstring>
4. #include<algorithm>
5. **using** **namespace** std;
6. **int** ans;
7. **bool** row[10],col[10];
8. **int** maze[10][10];
9. **int** n;
10. **bool** z[20],f[20];
11. **int** c[20];
12. **void** dd(**int** cur)//这一段对白皇后的搜索为白书中对八皇后问题求解的代码
13. {
14. **int** i,j;
15. **if**(cur==n)
16. ans++;
17. **else**
18. **for**(i=0;i<n;i++)
19. {
20. **int** ok=1;
21. **if**(maze[cur][i]==0)
22. **continue**;
23. c[cur]=i;
24. **for**(**int** j=0;j<cur;j++)
25. {
26. **if**(c[cur]==c[j]||cur-c[cur]==j-c[j]||cur+c[cur]==j+c[j])
27. {
28. ok=0;
29. **break**;
30. }
31. }
32. **if**(ok)
33. dd(cur+1);
34. }
35. }
36. **void** dfs(**int** count)
37. {
38. **if**(count==n)
39. {
40. dd(0);
41. **return** ;
42. }
43. **for**(**int** j=0;j<n;j++)
44. {
45. **if**(!row[count]&&!col[j]&&maze[count][j]==1&&!z[count+j]&&!f[n-1-j+count])
46. {
47. z[count+j]=**true**;
48. f[n-1-j+count]=**true**;
49. row[count]=**true**;
50. col[j]=**true**;
51. maze[count][j]=0;
52. dfs(count+1);
53. z[count+j]=**false**;
54. f[n-1-j+count]=**false**;
55. row[count]=**false**;
56. col[j]=**false**;
57. maze[count][j]=1;
58. }
59. }
60. **return** ;
61. }
62. **int** main()
63. {
64. cin>>n;
65. **for**(**int** i=0;i<n;i++)
66. **for**(**int** j=0;j<n;j++)
67. cin>>maze[i][j];
68. memset(row,0,**sizeof**(row));
69. memset(col,0,**sizeof**(col));
70. memset(z,0,**sizeof**(z));
71. memset(f,0,**sizeof**(f));
72. memset(c,-1,**sizeof**(c));
73. ans=0;
74. dfs(0);
75. cout<<ans<<endl;
77. }

5.  [**回形取数**](http://blog.csdn.net/buctyyzyn/article/details/43491109)

 问题描述

　　回形取数就是沿矩阵的边取数，若当前方向上无数可取或已经取过，则左转90度。一开始位于矩阵左上角，方向向下。

解题思路：

    直接按下，右，上，左的方向进行深搜即可。同样也可以用while循环实现，速度要比深搜快。

1. #include<cstdio>
2. #include<iostream>
3. #include<cstring>
4. #include<algorithm>
5. #define fr(i,n) for(int i=0;i<n;i++)
6. #define set(a,b) memset(a,b,sizeof(a))
7. **using** **namespace** std;
8. **int** m,n;
9. **int** maze[210][210];
10. **bool** vis[210][210];
11. **int** tt;
12. **int** ans[50000];
13. **bool** in(**int** x,**int** y)
14. {
15. **if**(x>=0&&x<m&&y>=0&&y<n)
16. **return** **true**;
17. **return** **false**;
18. }
19. **void** dfs(**int** x,**int** y,**int** dir)
20. {
21. vis[x][y]=1;
22. ans[tt++]=maze[x][y];
23. **if**(dir==0)
24. {
25. **if**(in(x+1,y)&&!vis[x+1][y])
26. dfs(x+1,y,0);
27. **else** **if**(in(x,y+1)&&!vis[x][y+1])
28. dfs(x,y+1,1);
29. }
30. **else** **if**(dir==1)
31. {
32. **if**(in(x,y+1)&&!vis[x][y+1])
33. dfs(x,y+1,1);
34. **else** **if**(in(x-1,y)&&!vis[x-1][y])
35. dfs(x-1,y,2);
36. }
37. **else** **if**(dir==2)
38. {
39. **if**(in(x-1,y)&&!vis[x-1][y])
40. dfs(x-1,y,2);
41. **else** **if**(in(x,y-1)&&!vis[x][y-1])
42. dfs(x,y-1,3);
43. }
44. **else** **if**(dir==3)
45. {
46. **if**(in(x,y-1)&&!vis[x][y-1])
47. dfs(x,y-1,3);
48. **else** **if**(in(x+1,y)&&!vis[x+1][y])
49. dfs(x+1,y,0);
50. }
51. }
52. **int** main()
53. {
55. cin>>m>>n;
56. tt=0;
57. set(vis,0);
58. set(ans,0);
60. fr(i,m)
61. fr(j,n)
62. cin>>maze[i][j];
63. dfs(0,0,0);
64. **int** t=0;
65. fr(i,m)
66. fr(j,n)
67. **if**(i==m-1&&j==n-1)
68. cout<<ans[t++]<<endl;
69. **else**
70. cout<<ans[t++]<<" ";
71. }

6.  [**2的次幂表示**](http://blog.csdn.net/buctyyzyn/article/details/43491453)

问题描述

　　任何一个正整数都可以用2进制表示，例如：137的2进制表示为10001001。  
　　将这种2进制表示写成2的次幂的和的形式，令次幂高的排在前面，可得到如下表达式：137=2^7+2^3+2^0  
　　现在约定幂次用括号来表示，即a^b表示为a（b）  
　　此时，137可表示为：2（7）+2（3）+2（0）  
　　进一步：7=2^2+2+2^0 （2^1用2表示）  
　　3=2+2^0   
　　所以最后137可表示为：2（2（2）+2+2（0））+2（2+2（0））+2（0）  
　　又如：1315=2^10+2^8+2^5+2+1  
　　所以1315最后可表示为：

　　2（2（2+2（0））+2）+2（2（2+2（0）））+2（2（2）+2（0））+2+2（0）

解题思路：仔细观察给出的几个例子会发现，处理二进制串的时候，遇到1就要进行递归，1之间遇到0就要填+号，进行递归时候要加括号，需要特判2^1,2^0

1. #include<cstdio>
2. #include<cstring>
3. #include<iostream>
4. #include<algorithm>
5. #include<cstdlib>
6. **using** **namespace** std;
7. **void** dfs(**int** n)
8. {
9. **if**(n==0)
10. {
11. cout<<"0";
12. **return** ;
13. }
14. string s;
15. **int** num=n;
16. **while**(num)
17. {
18. s+=num%2+'0';
19. num/=2;
20. }
21. reverse(s.begin(),s.end());
22. **int** flag=0;
23. **for**(**int** i=0;i<s.length();i++)
24. {
25. **if**(s[i]=='1')
26. {
27. **if**(flag)
28. cout<<"+";
29. flag=1;
30. **if**(s.length()-i-1==1)
31. {
32. cout<<"2";
33. }
34. **else**
35. {
36. cout<<"2";
37. cout<<"(";
38. dfs(s.length()-i-1);
39. cout<<")";
40. }
41. }
42. }
43. }
44. **int** main()
45. {
46. **int** num;
47. cin>>num;
48. string s;
49. **while**(num)
50. {
51. s+=num%2+'0';
52. num/=2;
53. }
54. reverse(s.begin(),s.end());
55. **int** flag=0;
56. **for**(**int** i=0;i<s.length();i++)
57. {
58. **if**(s[i]=='1')
59. {
60. **if**(flag)
61. cout<<"+";
62. flag=1;
63. **if**(s.length()-i-1==1)
64. {
65. cout<<"2";
66. }
67. **else**
68. {
69. cout<<"2";
70. cout<<"(";
71. dfs(s.length()-i-1);
72. cout<<")";
73. }
75. }
76. }
77. }

7. [导弹拦截-动态规划](http://blog.csdn.net/buctyyzyn/article/details/43525885)

问题描述

　　某国为了防御敌国的导弹袭击，发展出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。某天，雷达捕捉到敌国的导弹来袭。由于该系统还在试用阶段，所以只有一套系统，因此有可能不能拦截所有的导弹。

　　输入导弹依次飞来的高度（雷达给出的高度数据是不大于30000的正整数），计算这套系统最多能拦截多少导弹，如果要拦截所有导弹最少要配备多少套这种导弹拦截系统。

解题思路：

因为要求后面每一发炮弹都不能高于前一发，故问题就转化成了求最长不下降子序列，用n^2算法即可。

对于第二问，遇到高于前一项的，便要准备另外一套系统，而且要全部拦截，故求严格的最长上升子序列。

dp[i]表示以i位置结束的最长不下降子序列；

dps[i]表示以i位置结束的最长上升子序列；

注意初始化问题，对于每一个dp[i]，最少也会包括自身，所以都要初始化为1；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43525885)

1. #include<cstdio>
2. #include<cstring>
3. #include<algorithm>
4. #define max(a,b) ((a)>(b)?(a):(b))
5. **using** **namespace** std;
6. **int** main()
7. {
8. **int** dp[10010];
9. **int** dps[10010];
10. memset(dp,0,**sizeof**(dp));
11. memset(dps,0,**sizeof**(dps));
12. **int** n=0;
13. **int** num[10010];
14. **int** x;
15. **while**(~scanf("%d",&x))
16. num[++n]=x;
17. **for**(**int** i=1;i<=n;i++)
18. dp[i]=dps[i]=1;
19. **for**(**int** i=1;i<=n;i++)
20. {
21. **for**(**int** j=1;j<i;j++)
22. {
23. **if**(num[i]<=num[j])
24. {
25. dp[i]=max(dp[i],dp[j]+1);
26. }
27. **if**(num[i]>num[j])
28. {
29. dps[i]=max(dps[i],dps[j]+1);
30. }
31. }
32. }
33. **int** ans1=0,ans2=0;
34. **for**(**int** i=1;i<=n;i++)
35. {
36. ans1=max(ans1,dp[i]);
37. ans2=max(ans2,dps[i]);
38. }
39. printf("%d\n",ans1);
40. printf("%d\n",ans2);
41. }

9. [蓝桥杯 回文数](http://blog.csdn.net/buctyyzyn/article/details/43526447)

问题描述

　　若一个数（首位不为零）从左向右读与从右向左读都一样，我们就将其称之为回文数。

　　例如：给定一个10进制数56，将56加65（即把56从右向左读），得到121是一个回文数。

　　又如：对于10进制数87：

　　STEP1：87+78 = 165 STEP2：165+561 = 726

　　STEP3：726+627 = 1353 STEP4：1353+3531 = 4884

　　在这里的一步是指进行了一次N进制的加法，上例最少用了4步得到回文数4884。

　　写一个程序，给定一个N（2<=N<=10或N=16）进制数M（其中16进制数字为0-9与A-F），求最少经过几步可以得到回文数。

　　如果在30步以内（包含30步）不可能得到回文数，则输出“Impossible!”

输入格式

　　两行，N与M

输出格式

　　如果能在30步以内得到回文数，输出“STEP=xx”（不含引号），其中xx是步数；否则输出一行”Impossible!”（不含引号）

样例输入

9

87

问题描述

　　若一个数（首位不为零）从左向右读与从右向左读都一样，我们就将其称之为回文数。

　　例如：给定一个10进制数56，将56加65（即把56从右向左读），得到121是一个回文数。

　　又如：对于10进制数87：

　　STEP1：87+78 = 165 STEP2：165+561 = 726

　　STEP3：726+627 = 1353 STEP4：1353+3531 = 4884

　　在这里的一步是指进行了一次N进制的加法，上例最少用了4步得到回文数4884。

　　写一个程序，给定一个N（2<=N<=10或N=16）进制数M（其中16进制数字为0-9与A-F），求最少经过几步可以得到回文数。

　　如果在30步以内（包含30步）不可能得到回文数，则输出“Impossible!”

输入格式

　　两行，N与M

输出格式

　　如果能在30步以内得到回文数，输出“STEP=xx”（不含引号），其中xx是步数；否则输出一行”Impossible!”（不含引号）

样例输入

9

87

1. #include<iostream>
2. #include<cmath>
3. #include<algorithm>
4. **using** **namespace** std;
5. **long** **long** to10(**int** num,string s)
6. {
7. **long** **long** t=0;
8. **int** len=s.length();
9. **if**(num<=10)
10. {
11. **int** len=s.length();
12. **for**(**int** i=0;i<len;i++)
13. {
14. **if**(s[i]!='0')
15. {
16. t+=(s[i]-'0')\*pow(num,len-i-1);
17. }
18. }
19. }
20. **else**
21. {
22. **for**(**int** i=0;i<len;i++)
23. {
24. **if**(s[i]>'9')
25. {
26. t+=(s[i]-'A'+10)\*pow(num,len-i-1);
27. }
28. **else**
29. {
30. t+=(s[i]-'0')\*pow(num,len-i-1);
31. }
32. }
33. }
34. **return** t;
35. }
36. **bool** judge(**int** num,**long** **long** t)
37. {
38. string s;
39. **while**(t)
40. {
41. **int** cur=t%num;
42. t/=num;
43. **if**(cur<=9)
44. {
45. s+=cur+'0';
46. }
47. **else**
48. {
49. s+=cur-10+'A';
50. }
51. }
52. reverse(s.begin(),s.end());
53. **int** len=s.length();
54. **for**(**int** i=0;i<len/2;i++)
55. {
56. **if**(s[i]!=s[len-1-i])
57. **return** **false**;
58. }
59. **return** **true**;
60. }
61. **int** main()
62. {
63. **int** num;
64. string s;
65. cin>>num>>s;
66. **int** count=0;
67. **while**(count<31)
68. {
69. count++;
70. string ss=s;
71. reverse(ss.begin(),ss.end());
72. **long** **long** t1=to10(num,s);
73. **long** **long** t2=to10(num,ss);
74. **if**(judge(num,t1+t2))
75. **break**;
76. s.clear();
77. **long** **long** t=t1+t2;
78. **while**(t)
79. {
80. **int** cur=t%num;
81. t/=num;
82. **if**(cur<=9)
83. {
84. s+=cur+'0';
85. }
86. **else**
87. {
88. s+=cur-10+'A';
89. }
90. }
91. reverse(s.begin(),s.end());
92. }
93. **if**(count<31)
94. cout<<"STEP="<<count<<endl;
95. **else**
96. cout<<"Impossible!"<<endl;
97. }

10. [旅行家的预算](http://blog.csdn.net/buctyyzyn/article/details/43530405)

问题描述

　　一个旅行家想驾驶汽车以最少的费用从一个城市到另一个城市（假设出发时油箱是空的）。给定两个城市之间的距离D1、汽车油箱的容量C（以升为单位）、每升汽油能行驶的距离D2、出发点每升汽油价格P和沿途油站数N（N可以为零），油站i离出发点的距离Di、每升汽油价格Pi（i=1，2，……N）。计算结果四舍五入至小数点后两位。如果无法到达目的地，则输出“No Solution”。

输入格式

　　第一行为4个实数D1、C、D2、P与一个非负整数N；

　　接下来N行，每行两个实数Di、Pi。

输出格式

　　如果可以到达目的地，输出一个实数（四舍五入至小数点后两位），表示最小费用；否则输出“No Solution”（不含引号）。

样例输入

275.6 11.9 27.4 2.8 2

102.0 2.9

220.0 2.2

样例输出

26.95

解题思路：

贪心规则：

从当前点开始，再c\*d2的距离内，找第一个油价低于当前结点油价的站点

如果找到：

如果当前结点剩余油量可以抵达，更新下一个站点的剩余油量；

如果当前结点剩余油量不可以抵达，更新当前结点需要添加的油量；

如果找不到：

则在本站加满油，将相邻的结点当做下一个目的结点；-------》条件1

如果超过c\*d2，则不可以到达；

需要注意的是：判断找没找到时候应当注意用目的结点与当前结点的差值<c\*d2,判断。

如果单单用一个flag标志的话，当遇到最后一个结点，即目的地，你会直接跳到---》条件1---那个条件中，从而多加了油，使得结果偏大；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43530405)

1. #include<cstdio>
2. #include<algorithm>
3. **using** **namespace** std;
5. **struct** node
6. {
7. **double** dis;
8. **double** cost;
9. **double** has;
10. **double** add;
11. **int** id;
12. };
14. **int** main()
15. {
16. //freopen("test.txt","r",stdin);
17. **double** distance;
18. **double** c;
19. **double** v;
20. **double** scost;
21. **int** n;
22. node sta[10001];
23. node start;
25. scanf("%lf%lf%lf%lf%d",&distance,&c,&v,&scost,&n);
27. sta[0].dis=0;
28. sta[0].cost=scost;
29. sta[0].has=0;
30. sta[0].id=0;
31. sta[0].add=0;
32. sta[n+1].dis=distance;
33. sta[n+1].cost=100000000;
34. sta[n+1].id=n+1;
35. sta[n+1].has=0;
36. sta[n+1].add=0;
37. start=sta[0];
39. **for**(**int** i=1;i<=n;i++)
40. {
41. scanf("%lf%lf",&sta[i].dis,&sta[i].cost);
42. sta[i].has=0;
43. sta[i].id=i;
44. sta[i].add=0;
45. }
47. **bool** goal=**true**;
48. **double** ans=0;
49. **while**(start.id<n+1)
50. {
51. **double** len=c\*v;
52. **if**(sta[start.id+1].dis-start.dis>len)
53. {
54. goal=**false**;
55. printf("No Solution\n");
56. **break**;
57. }
59. **int** flag=**false**;
60. **int** k=1;
61. **while**(k<=n)
62. {
63. **if**(sta[k].id>start.id&&sta[k].dis-start.dis<len&&sta[k].cost<=start.cost)
64. {
65. flag=**true**;
66. **break**;
67. }
68. k++;
69. }
71. **if**(sta[k].dis-start.dis<len)
72. {
73. **if**(start.has\*v<sta[k].dis-start.dis)
74. {
75. start.add=(sta[k].dis-start.dis)/v-start.has;
76. start.has+=start.add;
77. sta[start.id]=start;
78. start=sta[k];
79. }
80. **else**
81. {
82. sta[k].has=start.has-(sta[k].dis-start.dis)/v;
83. start=sta[k];
84. }
85. }
86. **else**
87. {
88. k=start.id+1;
89. start.add=c-start.has;
90. start.has=c;
91. sta[start.id]=start;
92. sta[k].has=start.has-(sta[k].dis-start.dis)/v;
93. start=sta[k];
95. }
96. }
97. **if**(goal)
98. {
99. **for**(**int** i=0;i<=n;i++)
100. {
101. ans+=sta[i].add\*sta[i].cost;
102. }
103. printf("%.2lf\n",ans);
104. }

107. }

11. [集合运算](http://blog.csdn.net/buctyyzyn/article/details/43531209)

问题描述

　　给出两个整数集合A、B，求出他们的交集、并集以及B在A中的余集。

输入格式

　　第一行为一个整数n，表示集合A中的元素个数。

　　第二行有n个互不相同的用空格隔开的整数，表示集合A中的元素。

　　第三行为一个整数m，表示集合B中的元素个数。

　　第四行有m个互不相同的用空格隔开的整数，表示集合B中的元素。

　　集合中的所有元素均为int范围内的整数，n、m<=1000。

输出格式

　　第一行按从小到大的顺序输出A、B交集中的所有元素。

　　第二行按从小到大的顺序输出A、B并集中的所有元素。

　　第三行按从小到大的顺序输出B在A中的余集中的所有元素。

样例输入

5

1 2 3 4 5

5

2 4 6 8 10

样例输出

2 4

1 2 3 4 5 6 8 10

1 3 5

样例输入

4

1 2 3 4

3

5 6 7

样例输出

1 2 3 4 5 6 7

1 2 3 4

1. #include<cstdio>
2. #include<iostream>
3. #include<algorithm>
4. #define fr(a,b,c) for(int a=b;a<c;a++)
5. #define read(a) cin>>a
6. #include<set>
7. **using** **namespace** std;
8. **int** main()
9. {
10. **int** n,m;
11. read(n);
12. set<**int** > a,b,c;
13. **int** num;
14. fr(i,0,n)
15. {
16. read(num);
17. a.insert(num);
18. c.insert(num);
19. }
20. read(m);
21. fr(i,0,m)
22. {
23. read(num);
24. b.insert(num);
25. c.insert(num);
26. }
27. set<**int** >::iterator it;
28. **bool** flag=**false**;
29. **for**(it=a.begin();it!=a.end();it++)
30. {
32. **if**(b.find(\*it)!=b.end())
33. {
34. cout<<\*it<<" ";
35. flag=**true**;
36. }
37. }
38. **if**(flag)
39. {
40. cout<<endl;
41. flag=**false**;
42. }
43. **for**(it=c.begin();it!=c.end();it++)
44. {
45. cout<<(\*it)<<" ";
46. }
47. cout<<endl;
48. **for**(it=a.begin();it!=a.end();it++)
49. {
50. **if**(b.find(\*it)==b.end())
51. {
52. flag=**true**;
53. cout<<\*it<<" ";
54. }
55. }
56. **if**(flag)
57. {
58. cout<<endl;
59. flag=**false**;
60. }
62. }

12. [摆动序列](http://blog.csdn.net/buctyyzyn/article/details/43550419)

问题描述

　　如果一个序列满足下面的性质，我们就将它称为摆动序列：  
　　1. 序列中的所有数都是不大于k的正整数；  
　　2. 序列中至少有两个数。  
　　3. 序列中的数两两不相等；  
　　4. 如果第i – 1个数比第i – 2个数大，则第i个数比第i – 2个数小；如果第i – 1个数比第i – 2个数小，则第i个数比第i – 2个数大。  
　　比如，当k = 3时，有下面几个这样的序列：  
　　1 2  
　　1 3  
　　2 1  
　　2 1 3  
　　2 3  
　　2 3 1  
　　3 1  
　　3 2

　　一共有8种，给定k，请求出满足上面要求的序列的个数。

解题思路：

说的是动态规划的题目，但最终还是没有想出来转移方程，如果有大牛知道，可以留言，谢谢。

给定K，则序列最多K位，可以递归搜索每一层，当满足条件时候，ans++即可。

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43550419)[在CODE上查看代码片](https://code.csdn.net/snippets/598963)

1. #include<cstdio>
2. #include<cstring>
3. #include<iostream>
4. #include<algorithm>
5. **using** **namespace** std;
6. #define read(a) cin>>a
7. #define set(a,b) memset(a,b,sizeof(a))
8. **bool** vis[21];
9. **int** sans[21];
10. **int** n;
11. **int** ans;
12. **void** dfs(**int** count)
13. {
14. **if**(count>n)
15. **return** ;
16. **for**(**int** i=1;i<=n;i++)
17. {
18. **if**(!vis[i])
19. {
20. sans[count]=i;
21. **if**(count==1)
22. {
23. vis[i]=**true**;
24. dfs(count+1);
25. vis[i]=**false**;
26. }
27. **else** **if**(count==2)
28. {
29. ans++;
30. vis[i]=**true**;
31. dfs(count+1);
32. vis[i]=**false**;
33. }
34. **else** **if**(count>2)
35. {
36. **if**((sans[count-1]>sans[count-2]&&sans[count]<sans[count-2])||(sans[count-1]<sans[count-2]&&sans[count]>sans[count-2]))
37. {
38. ans++;
39. vis[i]=**true**;
40. dfs(count+1);
41. vis[i]=**false**;
42. }
43. }
44. }
45. }
46. }
47. **int** main()
48. {
50. read(n);
51. set(vis,0);
52. ans=0;
53. dfs(1);
54. printf("%d\n",ans);
55. }

13.  [**瓷砖铺放**](http://blog.csdn.net/buctyyzyn/article/details/43551181)

问题描述

　　有一长度为N(1<=Ｎ<=10)的地板，给定两种不同瓷砖：一种长度为1，另一种长度为2，数目不限。要将这个长度为N的地板铺满，一共有多少种不同的铺法？

　　例如，长度为4的地面一共有如下5种铺法：

　　4=1+1+1+1

　　4=2+1+1

　　4=1+2+1

　　4=1+1+2

　　4=2+2

　　编程用递归的方法求解上述问题。

输入格式

　　只有一个数N，代表地板的长度

输出格式

　　输出一个数，代表所有不同的瓷砖铺放方法的总数

样例输入

4

样例输出

5

1. #include<cstdio>
2. #include<cstring>
3. #include<iostream>
4. #include<algorithm>
5. **using** **namespace** std;
6. #define read(a) cin>>a
7. #define set(a,b) memset(a,b,sizeof(a))
8. **int** ans;
9. **int** n;
10. **void** dfs(**int** num,**int** a)
11. {
12. **if**(a>num)
13. **return** ;
14. **if**(a==num)
15. {
16. ans++;
17. **return** ;
18. }
19. dfs(num,a+1);
20. dfs(num,a+2);
21. }
22. **int** main()
23. {
24. read(n);
25. ans=0;
26. dfs(n,0);
27. cout<<ans<<endl;
28. }

14.  [**入学考试--采药问题**](http://blog.csdn.net/buctyyzyn/article/details/43554841)

问题描述

　　辰辰是个天资聪颖的孩子，他的梦想是成为世界上最伟大的医师。为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。医师把他带到一个到处都是草药的山洞里对他说：“孩子，这个山洞里有一些不同的草药，采每一株都需要一些时间，每一株也有它自身的价值。我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”

　　如果你是辰辰，你能完成这个任务吗？

解题思路：

经典的0-1背包题目，对于每一种物品，只有拿或不拿俩种策略；

对于0-1背包的俩行核心代码，很多人不明白为什么第二个for循环要倒着递推，原因在于递推dp[j]的时候，需要用到dp[j-w[i]],而j-w[i]<j,如果正着递推，在i物品下，调用dp[j-w[i]]的值，不能保证它是拿前几种物品的结果，因为你已经修改过了！！！！因此倒着递推可以保证dp[j-w[i]]的值为拿前面物品得到的结果。

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43554841)

1. #include<cstdio>
2. #include<cstring>
3. #include<iostream>
4. #include<algorithm>
5. **using** **namespace** std;
6. #define fr(a,b,c) for(int a=b;a<c;a++)
7. #define max(a,b) ((a)>(b)?(a):(b))
8. #define read(a) cin>>a
9. #define set(a,b) memset(a,b,sizeof(a))
10. **int** main()
11. {
12. **int** v,n;
13. read(v);
14. read(n);
15. **int** w[110],val[110];
16. **int** dp[1010];
17. set(w,0);
18. set(val,0);
19. set(dp,0);
20. fr(i,0,n)
21. {
22. read(w[i]);
23. read(val[i]);
24. }
25. **for**(**int** i=0;i<n;i++)
26. **for**(**int** j=v;j>=w[i];j--)
27. {
28. dp[j]=max(dp[j],dp[j-w[i]]+val[i]);
29. }
30. cout<<dp[v]<<endl;
31. }

15.  [**开心的金明**](http://blog.csdn.net/buctyyzyn/article/details/43555069)

问题描述

　　金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间他自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说：“你的房间需要购买哪些物品，怎 么布置，你说了算，只要不超过N元钱就行”。今天一早金明就开始做预算，但是他想买的东西太多了，肯定会超过妈妈限定的N元。于是，他把每件物品规定了一 个重要度，分为5等：用整数1~5表示，第5等最重要。他还从因特网上查到了每件物品的价格（都是整数元）。他希望在不超过N元（可以等于N元）的前提 下，使每件物品的价格与重要度的乘积的总和最大。  
　　设第j件物品的价格为v[j]，重要度为w[j]，共选中了k件物品，编号依次为 j1，j2，……，jk，则所求的总和为：  
　　v[j1]\*w[j1]+v[j2]\*w[j2]+ …+v[jk]\*w[jk]。（其中\*为乘号）

　　请 你帮助金明设计一个满足要求的购物单。

解题思路：

0-1背包问题。

状态转移方程dp[j]=max(dp[j],dp[j-w[i]]+w[i]\*val[i]);

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43555069)

1. #include<cstdio>
2. #include<cstring>
3. #include<iostream>
4. #include<algorithm>
5. **using** **namespace** std;
6. #define fr(a,b,c) for(int a=b;a<c;a++)
7. #define max(a,b) ((a)>(b)?(a):(b))
8. #define read(a) cin>>a
9. #define set(a,b) memset(a,b,sizeof(a))
10. **int** main()
11. {
12. **int** v,n;
13. read(v);
14. read(n);
15. **int** w[110],val[110];
16. **int** dp[30100];
17. set(w,0);
18. set(val,0);
19. set(dp,0);
20. fr(i,0,n)
21. {
22. read(w[i]);
23. read(val[i]);
24. }
25. **for**(**int** i=0;i<n;i++)
26. **for**(**int** j=v;j>=w[i];j--)
27. {
28. dp[j]=max(dp[j],dp[j-w[i]]+val[i]\*w[i]);
29. }
30. cout<<dp[v]<<endl;
31. }

16.  [**安慰奶牛**](http://blog.csdn.net/buctyyzyn/article/details/43817935)

问题描述：

Farmer John变得非常懒，他不想再继续维护供奶牛之间供通行的道路。道路被用来连接N个牧场，牧场被连续地编号为1到N。每一个牧场都是一个奶牛的家。FJ计划除去P条道路中尽可能多的道路，但是还要保持牧场之间 的连通性。你首先要决定那些道路是需要保留的N-1条道路。第j条双向道路连接了牧场Sj和Ej(1 <= Sj <= N; 1 <= Ej <= N; Sj != Ej)，而且走完它需要Lj的时间。没有两个牧场是被一条以上的道路所连接。奶牛们非常伤心，因为她们的交通系统被削减了。你需要到每一个奶牛的住处去安慰她们。每次你到达第i个牧场的时候(即使你已经到过)，你必须花去Ci的时间和奶牛交谈。你每个晚上都会在同一个牧场(这是供你选择的)过夜，直到奶牛们都从悲伤中缓过神来。在早上 起来和晚上回去睡觉的时候，你都需要和在你睡觉的牧场的奶牛交谈一次。这样你才能完成你的 交谈任务。假设Farmer John采纳了你的建议，请计算出使所有奶牛都被安慰的最少时间。

输入格式

第1行包含两个整数N和P。

接下来N行，每行包含一个整数Ci。

接下来P行，每行包含三个整数Sj, Ej和Lj。

输出格式

输出一个整数, 所需要的总时间(包含和在你所在的牧场的奶牛的两次谈话时间)。

样例输入

5 7  
10  
10  
20  
6  
30  
1 2 5  
2 3 5  
2 4 12  
3 4 17  
2 5 15  
3 5 6  
4 5 12

解题思路：

1：因为结点数量较大，prim算法势必要超时，故选用kruskl.

2：每次走过一个结点，都要停留Ci时间，实际中走的时候，在最小生成树走的时候实际上要走俩次，因此边权为Lj\*2+C[s]+C[e]；题目中要求最后要回到原点，所以还要加一个出发点的权值，出发点当然一定要选最小的！

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43817935)

1. #include<cstdio>
2. #include<cstring>
3. #include<iostream>
4. #include<algorithm>
5. **using** **namespace** std;
7. **struct** Edge
8. {
9. **int** x;
10. **int** y;
11. **int** cost;
12. }edge[100001];
13. **int** set[100001];
14. **int** val[100001];
15. **inline** **int** find(**int** x)
16. {
17. **int** i,j,r;
18. r=x;
19. **while**(set[r]!=r)
20. r=set[r];
21. i=x;
22. **while**(i!=r)
23. {
24. j=set[i];
25. set[i]=r;
26. i=j;
27. }
28. **return** r;
29. }
30. **inline** **void** merge(**int** x,**int** y)
31. {
32. **int** t=find(x);
33. **int** h=find(y);
34. **if**(t<h)
35. set[h]=t;
36. **else**
37. set[t]=h;
38. }
40. **bool** cmp(**const** Edge& a,**const** Edge& b)
41. {
42. **return** a.cost<b.cost;
43. }
45. **void** init(**int** n)
46. {
47. **for**(**int** i=1;i<=n;i++)
48. set[i]=i;
49. }
50. **int** kruskal(**int** n,**int** m)
51. {
52. **int** i,num,sum,from,to;
53. sort(edge,edge+m,cmp);
54. sum=num=0;
55. **for**(i=0;i<m;i++)
56. {
57. from = find(edge[i].x);
58. to = find(edge[i].y);
59. **if**(from != to)
60. {
61. merge(from,to);
62. sum+=edge[i].cost;
63. num++;
64. }
65. **if**(num==n-1)
66. **break**;
67. }
68. **if**(num<n-1)
69. **return** -1;
70. **else**
71. **return** sum;
72. }
74. **int** main()
75. {
76. **int** i,n,m,k;
77. **int** minn=0x3f3f3f3f;
78. memset(val,0,**sizeof**(val));
79. **while**(scanf("%d%d",&n,&m)!=EOF)
80. {
81. init(n);
82. **for**(i=1;i<=n;i++)
83. {
84. scanf("%d",&val[i]);
85. minn=min(minn,val[i]);
86. }
87. **for**(i=0;i<m;i++)
88. {
89. scanf("%d%d%d",&edge[i].x,&edge[i].y,&edge[i].cost);
90. edge[i].cost+=edge[i].cost;
91. edge[i].cost+=val[edge[i].x]+val[edge[i].y];
92. }
93. k=kruskal(n,m)+minn;
94. **if**(k==-1)
95. printf("no\n");
96. **else**
97. printf("%d\n",k);
98. }
99. }

17.  [**POJ 3481 Double Queue**](http://blog.csdn.net/buctyyzyn/article/details/43830439)

题意：

有3种操作：

1 K P，把权值为K，优先级P的数据存在查询的表里；

2，返回优先级最大的权值，并删除该项

3，返回优先级最小的权值，并删除该项

思路：

刚学了SBT，用来练手了，我用map存储了优先级对应的答案，700+mm，感觉时间太长了。。。。。。

1. #include <stdio.h>
2. #include <iostream>
3. #include <string.h>
4. #include <algorithm>
5. #include <math.h>
6. #include<map>
7. **using** **namespace** std;
8. **const** **int** MAXN=111111;
9. **struct** SBT
10. {
11. **int** left,right,size,key;
12. **int** val;
13. **void** Init()
14. {
15. left=right=0;
16. size=1;
17. }
18. }tree[MAXN];
19. **int** tot,root;
20. **void** left\_rotate(**int** &x)//左旋
21. {
22. **int** y=tree[x].right;
23. tree[x].right=tree[y].left;
24. tree[y].left=x;
25. tree[y].size=tree[x].size;
26. tree[x].size=tree[tree[x].left].size+tree[tree[x].right].size+1;
27. x=y;
28. }
29. **void** right\_rotate(**int** &x)//右旋
30. {
31. **int** y=tree[x].left;
32. tree[x].left=tree[y].right;
33. tree[y].right=x;
34. tree[y].size=tree[x].size;
35. tree[x].size=tree[tree[x].left].size+tree[tree[x].right].size+1;
36. x=y;
37. }
38. **void** maintain(**int** &x,**int** flag)
39. {
40. **if**(flag==0)
41. {
42. **if**(tree[tree[tree[x].left].left].size > tree[tree[x].right].size)
43. right\_rotate(x);
44. **else** **if**(tree[tree[tree[x].left].right].size > tree[tree[x].right].size)
45. left\_rotate(tree[x].left),right\_rotate(x);
46. **else** **return**;
47. }
48. **else**
49. {
50. **if**(tree[tree[tree[x].right].right].size > tree[tree[x].left].size)
51. left\_rotate(x);
52. **else** **if**(tree[tree[tree[x].right].left].size > tree[tree[x].left].size)
53. right\_rotate(tree[x].right),left\_rotate(x);
54. **else** **return**;
55. }
56. maintain(tree[x].left,0);
57. maintain(tree[x].right,1);
58. maintain(x,0);
59. maintain(x,1);
60. }
61. //插入元素，相同元素放在右子树中
62. **void** insert(**int** &x,**int** key)
63. {
64. **if**(x==0)
65. {
66. x=++tot;
67. tree[x].Init();
68. tree[x].key=key;
69. }
70. **else**
71. {
72. tree[x].size++;
73. **if**(key<tree[x].key)insert(tree[x].left,key);
74. **else** insert(tree[x].right,key);
75. maintain(x,key>=tree[x].key);
76. }
77. }
78. //删除key值的元素
79. **int** del(**int** &x,**int** key)
80. {
81. **if**(!x)**return** 0;
82. tree[x].size--;
83. **if**(key==tree[x].key || (key<tree[x].key&&tree[x].left==0)
84. || (key>tree[x].key&&tree[x].right==0))
85. {
86. **if**(tree[x].left && tree[x].right)
87. {
88. **int** p=del(tree[x].left,key+1);
89. tree[x].key=tree[p].key;
90. **return** p;
91. }
92. **else**
93. {
94. **int** p=x;
95. x=tree[x].left+tree[x].right;
96. **return** p;
97. }
98. }
99. **else** **return** del(key<tree[x].key?tree[x].left:tree[x].right,key);
100. }
102. //取最大值
103. //返回最大值的节点编号
104. **int** Get\_Max(**int** x)
105. {
106. **while**(tree[x].right)x=tree[x].right;
107. **return** x;
108. }
109. //取最小值
110. //返回最小值的节点编号
111. **int** Get\_Min(**int** x)
112. {
113. **while**(tree[x].left)x=tree[x].left;
114. **return** x;
115. }
117. **int** main()
118. {
119. root = tot = 0;
120. **int** k,aa,bb;
121. map<**int**,**int**> ans;
122. **while**(scanf("%d",&k),k)
123. {
124. **if**(k==2)
125. {
126. **if**(root==0)
127. printf("0\n");
128. **else**
129. {
130. **int** t=tree[Get\_Max(root)].key;
131. printf("%d\n",ans[t]);
132. del(root,t);
133. }
134. }
135. **if**(k==3)
136. {
137. **if**(root==0)
138. printf("0\n");
139. **else**
140. {
141. **int** t=tree[Get\_Min(root)].key;
142. printf("%d\n",ans[t]);
143. del(root,t);
144. }
145. }
146. **if**(k==1)
147. {
148. scanf("%d%d",&aa,&bb);
149. ans[bb]=aa;
150. //printf("%d\n",ans[bb]);
151. insert(root,bb);
152. }
153. }
154. }

18. [进制转换](http://blog.csdn.net/buctyyzyn/article/details/43835527)

问题描述

　　我们可以用这样的方式来表示一个十进制数： 将每个阿拉伯数字乘以一个以该数字所处位置的（值减１）为指数，以１０为底数的幂之和的形式。例如：１２３可表示为 １＊１０２＋２＊１０１＋３＊１００这样的形式。  
　　与之相似的，对二进制数来说，也可表示成每个二进制数码乘以一个以该数字所处位置的（值－１）为指数，以２为底数的幂之和的形式。一般说来，任何一个正整数Ｒ或一个负整数－Ｒ都可以被选来作为一个数制系统的基数。如果是以Ｒ或－Ｒ为基数，则需要用到的数码为 ０，１，．．．．Ｒ－１。例如，当Ｒ＝７时，所需用到的数码是０，１，２，３，４，５和６，这与其是Ｒ或－Ｒ无关。如果作为基数的数绝对值超过１０，则为了表示这些数码，通常使用英文字母来表示那些大于９的数码。例如对１６进制数来说，用Ａ表示１０，用Ｂ表示１１，用Ｃ表示１２，用Ｄ表示１３，用Ｅ表示１４，用Ｆ表示１５。  
　　在负进制数中是用－Ｒ 作为基数，例如－１５（十进制）相当于１１０００１（－２进制），并且它可以被表示为２的幂级数的和数：  
　　１１０００１＝１＊（－２）５＋１＊（－２）４＋０＊（－２）３＋０＊（－２）２＋  
　　０＊（－２）１ ＋１＊（－２）０

　　 设计一个程序，读入一个十进制数和一个负进制数的基数, 并将此十进制数转换为此负进制下的数： －Ｒ∈｛－２，－３，－４，．．．，－２０｝

解题思路：

负数进制一样。每次取的余数保证在0~-m-1之间。（例如m=-16,则余数应该在0~15）就可以直接输出。　所以用系统的“mod”运算符的时候必须注意检查是不是在该范围（可能在m+1~0），否则就调整。调整的方法是：

if 余数<0 then

begin

余数=余数-m;

商=商+1;

end;

1. #include<cstdio>
2. #include<algorithm>
3. #include<iostream>
4. #include<string>
5. **using** **namespace** std;
6. **void** change(**int** num,**int** k)
7. {
8. string s;
9. **while**(num)
10. {
11. **int** t=num%k;
12. **int** tt=num/k;
13. **if**(t<0)
14. {
15. t-=k;
16. tt++;
17. }
18. **if**(t>9)
19. {
20. s+='A'+t-10;
21. }
22. **else**
23. s+='0'+t;
24. num=tt;
25. }
26. reverse(s.begin(),s.end());
27. cout<<s;
28. }
29. **int** main()
30. {
31. **int** t;
32. **int** k;
33. cin>>t>>k;
34. cout<<t<<"=";
35. change(t,k);
36. cout<<"(base"<<k<<")"<<endl;
37. }

19. [乘积最大](http://blog.csdn.net/buctyyzyn/article/details/43851695)

问题描述

　　今年是国际数学联盟确定的“2000——世界数学年”，又恰逢我国著名数学家华罗庚先生诞辰90周年。在华罗庚先生的家乡江苏金坛，组织了一场别开生面的数学智力竞赛的活动，你的一个好朋友XZ也有幸得以参加。活动中，主持人给所有参加活动的选手出了这样一道题目：  
  
　　设有一个长度为N的数字串，要求选手使用K个乘号将它分成K+1个部分，找出一种分法，使得这K+1个部分的乘积能够为最大。  
  
　　同时，为了帮助选手能够正确理解题意，主持人还举了如下的一个例子：  
  
　　有一个数字串：312， 当N=3，K=1时会有以下两种分法：  
  
　　3\*12=36  
　　31\*2=62  
  
　　这时，符合题目要求的结果是：31\*2=62  
  
　　现在，请你帮助你的好朋友XZ设计一个程序，求得正确的答案。

   输入格式

　　程序的输入共有两行：  
　　第一行共有2个自然数N，K（6≤N≤40，1≤K≤6）  
　　第二行是一个长度为N的数字串。

解题思路：

动态规划的经典题目。

i从0开始；

ans[i][j]表示长度为i+1的数字串插入j个\*号能够获得的最大的乘积；

ans[i][0]显然是长度为i+1的数字串所对应的数字；

状态转移方程：

利用最优子结构的性质，如果在k个字符后插入第j个\*取得最大值，则ans[k][j-1]也是最大，因此只需要在确定i,j的情况下，只需要枚举最后一个\*号的位置即可

ans[i][j]=max(ans[i][j],ans[k][j-1]\*(k+1个位置起，i-k个长度字符串所对应的数字)；0<=k<i;1<=j<=min(i,\*最大的个数)；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43851695)

1. #include<cstdio>
2. #include<cstring>
3. #include<iostream>
4. #include<string>
5. #include<algorithm>
6. **using** **namespace** std;
7. **int** change(string a)
8. {
9. **int** ans=a[0]-'0';
10. **int** len=a.length();
11. **for**(**int** i=1;i<len;i++)
12. {
13. ans=ans\*10+a[i]-'0';
14. }
15. **return** ans;
16. }
17. **int** main()
18. {
19. **int** m,n;
20. cin>>m>>n;
21. string s;
22. cin>>s;
23. **long** **long** ans[45][7];
24. memset(ans,0,**sizeof**(ans));
25. **for**(**int** i=1;i<=m;i++)
26. ans[i-1][0]=change(s.substr(0,i));
28. **for**(**int** i=0;i<m;i++)
29. {
30. **int** t=min(i,n);
31. **for**(**int** j=1;j<=t;j++)
32. {
33. **for**(**int** k=0;k<i;k++)
34. {
35. **long** **long** tt=change(s.substr(k+1,i-k));
36. ans[i][j]=max(ans[i][j],ans[k][j-1]\*tt);
37. }
38. }
39. }
40. cout<<ans[m-1][n];
41. }

20. [单词接龙](http://blog.csdn.net/buctyyzyn/article/details/43852605)

问题描述

　　单词接龙是一个与我们经常玩的成语接龙相类似的游戏，现在我们已知一组单词，且给定一个开头的字母，要求出以这个字母开头的最长的“龙”（每个单词都最多在“龙”中出现两次），在两个单词相连时，其重合部分合为一部分，例如 beast和astonish，如果接成一条龙则变为beastonish，另外相邻的两部分不能存在包含关系，例如at 和 atide 间不能相连。

输入格式

　　输入的第一行为一个单独的整数n (n<=20)表示单词数，以下n 行每行有一个单词，输入的最后一行为一个单个字符，表示“龙”开头的字母。你可以假定以此字母开头的“龙”一定存在.

输出格式

　　只需输出以此字母开头的最长的“龙”的长度

样例输入

5  
at  
touch  
cheat  
choose  
tact  
a

样例输出

23

样例说明

　　连成的“龙”为atoucheatactactouchoose

解题思路：

对首字母包含给出的字母的单词进行深搜，枚举其不包含单词首字母的后缀，遇到单词的前缀含有模式串后缀的，进行深搜，该过程中根据模式串的长度不断的更新ans的最大值

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43852605)

1. #include<cstdio>
2. #include<cstring>
3. #include<string>
4. #include<algorithm>
5. #include<iostream>
6. **using** **namespace** std;
7. string word[21];
8. string s;
9. **int** n;
10. **int** vis[21];
11. **int** ans;
12. **void** dfs(string t)
13. {
14. **int** len=t.length();
15. ans=max(ans,len);
16. **for**(**int** i=1;i<len;i++)
17. {
18. string tt=t.substr(i,len-i);
19. **for**(**int** j=0;j<n;j++)
20. {
21. **int** kk=word[j].length();
22. **if**(vis[j]<2)
23. {
24. **if**(word[j].find(tt)==0)
25. {
26. **if**(word[j]==tt)
27. **continue**;
28. vis[j]++;
29. string cur=t;//注意此处用临时变量替换t，这样可以保证在这一层的深搜中t没有被修改
30. dfs(cur.replace(i,tt.length(),word[j]));
31. vis[j]--;
32. }
33. }
34. }
35. }
36. }
37. **int** main()
38. {
39. ans=0;
40. memset(vis,0,**sizeof**(vis));
41. cin>>n;
42. **for**(**int** i=0;i<n;i++)
43. cin>>word[i];
44. cin>>s;
45. **for**(**int** i=0;i<n;i++)
46. **if**(word[i].find(s)==0)
47. {
48. vis[i]++;
49. dfs(word[i]);
50. vis[i]--;
51. }
52. cout<<ans<<endl;
53. }

# 21.[方格取数](http://blog.csdn.net/buctyyzyn/article/details/43863611)

问题描述

　　设有N\*N的方格图(N<=10),我们将其中的某些方格中填入正整数,而其他的方格中则放入数字0。  
　　某人从图的左上角的A 点(1,1)出发，可以向下行走，也可以向右走，直到到达右下角的B点(N,N)。在走过的路上，他可以取走方格中的数（取走后的方格中将变为数字0）。  
　　此人从A点到B 点共走两次，试找出2条这样的路径，使得取得的数之和为最大。

输入格式

　　输入的第一行为一个整数N（表示N\*N的方格图），接下来的每行有三个整数，前两个表示位置，第三个数为该位置上所放的数。一行单独的0表示输入结束。

输出格式

　　只需输出一个整数，表示2条路径上取得的最大的和。

样例输入

8  
2 3 13  
2 6 6  
3 5 7  
4 4 14  
5 2 21  
5 6 4  
6 3 15  
7 2 14  
0 0 0

样例输出

67

解题思路：

多线程同时dp。当做俩个人同时在矩阵中走；

i1,j1,表示第一个人的位置；

i2,j2，表示第二个人的位置；

dp[i1][j1][i2][j2]表示俩个人在对应的方格中时候取得的最大的数；

到达i1,j1有(i1-1,j1)(i1,j1-1)俩种状态；

到达i2,j2有(i2-1,j2)(i2,j2-1)俩种状态；

因此dp[i1][j1][i2][j2]就是这四种dp[][][][]里的最大值加上对应maze[i1][j1],maze[i2][j2]中的值即可，但是，要注意，如果俩个人站在同一个位置，则只加一个maze即可，因为一个人取走后该位置就变为0了。

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43863611)[在CODE上查看代码片](https://code.csdn.net/snippets/605734)

1. #include<cstdio>
2. #include<iostream>
3. #include<cstring>
4. #include<string>
5. #include<algorithm>
6. **using** **namespace** std;
7. **int** main()
8. {
9. **int** n;
10. cin>>n;
11. **long** **long** maze[11][11];
12. **long** **long** dp[11][11][11][11];
13. memset(maze,0,**sizeof**(maze));
14. **int** a,b,c;
15. **while**(cin>>a>>b>>c)
16. {
17. **if**(a&&b&&c)
18. maze[a][b]=c;
19. **else**
20. **break**;
21. }
22. memset(dp,0,**sizeof**(dp));
23. **for**(**int** i1=1;i1<=n;i1++)
24. {
25. **for**(**int** j1=1;j1<=n;j1++)
26. {
27. **for**(**int** i2=1;i2<=n;i2++)
28. {
29. **for**(**int** j2=1;j2<=n;j2++)
30. {
31. dp[i1][j1][i2][j2]=max(max(dp[i1-1][j1][i2-1][j2],dp[i1-1][j1][i2][j2-1]),max(dp[i1][j1-1][i2-1][j2],dp[i1][j1-1][i2][j2-1]));
32. dp[i1][j1][i2][j2]+=maze[i1][j1];
33. **if**(i1!=i2||j1!=j2)
34. dp[i1][j1][i2][j2]+=maze[i2][j2];
35. }
36. }
37. }
38. }
39. cout<<dp[n][n][n][n]<<endl;
40. }

22.  [**求先序排列**](http://blog.csdn.net/buctyyzyn/article/details/43866661)

问题描述

　　  给出一棵二叉树的中序与后序排列。求出它的先序排列。（约定树结点用不同的大写字母表示，长度<=8）。

         输入格式

　　  两行，每行一个字符串，分别表示中序和后序排列

         输出格式

　　  一个字符串，表示所求先序排列

         样例输入

BADC  
BDCA

样例输出

ABCD

解题思路：

后序遍历的最后一位是根，因此可以在中序遍历中找到后序遍历的根，然后递归右子树，左子树即可

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43866661)

1. #include<cstdio>
2. #include<iostream>
3. #include<cstring>
4. #include<string>
5. #include<algorithm>
6. **using** **namespace** std;
7. **struct** node{
8. **char** data;
9. **struct** node \*lc,\*rc;
10. };
11. **bool** vis[26];
12. **void** dfs(node \*&T,string a,string b)
13. {
14. **int** lb=b.length();
15. **int** la=a.length();
16. **if**(lb>0&&la>0)
17. {
18. T=(node \*)malloc(**sizeof**(node));
19. **int** t=lb-1;
20. **while**(t)
21. {
22. **if**(!vis[b[t]-'A'])
23. **break**;
24. **else**
25. t--;
26. }
27. **if**(t<0) **return** ;
28. T->data=b[t];
29. vis[b[t]-'A']=**true**;
30. dfs(T->rc,a.substr(a.find(b[t])+1,la-a.find(b[t])-1),b.substr(0,t));
31. dfs(T->lc,a.substr(0,a.find(b[t])),b.substr(0,t));
32. }
33. **else**
34. T=NULL;
35. }
36. **void** visit(node \*T)
37. {
38. **if**(T!=NULL)
39. {
40. cout<<T->data;
41. visit(T->lc);
42. visit(T->rc);
43. }
44. }
45. **int** main()
46. {
47. string s1,s2;
48. cin>>s1>>s2;
49. memset(vis,0,**sizeof**(vis));
50. **int** l1=s1.length(),l2=s2.length();
51. node \*T;
52. dfs(T,s1,s2);
53. visit(T);
54. }

24. [装箱问题](http://blog.csdn.net/buctyyzyn/article/details/43866693)

问题描述

　　有一个箱子容量为V（正整数，0＜＝V＜＝20000），同时有n个物品（0＜n＜＝30），每个物品有一个体积（正整数）。  
　　要求n个物品中，任取若干个装入箱内，使箱子的剩余空间为最小。

输入格式

　　第一行为一个整数，表示箱子容量；  
　　第二行为一个整数，表示有n个物品；  
　　接下来n行，每行一个整数表示这n个物品的各自体积。

输出格式

　　一个整数，表示箱子剩余空间。

   解题思路：

0-1背包问题；物品的体积既是费用也是价值，最后用v减去dp[v]即可；

状态转移方程dp[j]=max(dp[j],dp[j-num[i]]+num[i]),0=<i<n;

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43866693)[在CODE上查看代码片](https://code.csdn.net/snippets/605913)

1. #include<cstdio>
2. #include<cstring>
3. #define max(a,b) ((a)>(b)?(a):(b))
4. **int** main()
5. {
6. **int** v,n;
7. **int** num[31];
8. scanf("%d",&v);
9. scanf("%d",&n);
10. **for**(**int** i=0;i<n;i++)
11. scanf("%d",&num[i]);
12. **int** dp[20010];
13. memset(dp,0,**sizeof**(dp));
14. **for**(**int** i=0;i<n;i++)
15. {
16. **for**(**int** j=v;j>=num[i];j--)
17. {
18. dp[j]=max(dp[j],dp[j-num[i]]+num[i]);
19. }
20. }
21. printf("%d\n",v-dp[v]);
22. }

25. [数的划分](http://blog.csdn.net/buctyyzyn/article/details/43870147)

问题描述

　　将整数n分成k份，且每份不能为空，任意两份不能相同(不考虑顺序)。  
　　例如：n=7，k=3，下面三种分法被认为是相同的。  
　　1，1，5; 1，5，1; 5，1，1;  
　　问有多少种不同的分法。

输入格式

　　n，k

输出格式

　　一个整数，即不同的分法

样例输入

7 3

样例输出

4 {四种分法为：1，1，5;1，2，4;1，3，3;2，2，3;}

解题思路：

dp[n][k]表示把划分为k个数的方案种数；

有俩种情况，不包含1：

则可以先找出k个1放在每一份中，然后把剩下的n-k分成k份即可；

包含1：

则先把1拿出来当做单独的一份，则把n-1划分成k-1份即可；

状态转移方程为：

dp[i][j]=dp[i-j][j]+dp[i-1][j-1];         i>j

dp[i][j]=1;                               i=j

dp[i][j]=0;                               i<j

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43870147)[在CODE上查看代码片](https://code.csdn.net/snippets/606023)

1. #include<cstdio>
2. #include<cstring>
3. **int** main()
4. {
5. **int** n,k;
6. scanf("%d%d",&n,&k);
7. **long** **long** dp[201][7];
8. memset(dp,0,**sizeof**(dp));
9. **for**(**int** i=1;i<=n;i++)
10. {
11. **for**(**int** j=1;j<=k;j++)
12. **if**(i<j)
13. dp[i][j]=0;
14. **else** **if**(i==j)
15. dp[i][j]=1;
16. **else**
17. dp[i][j]=dp[i-j][j]+dp[i-1][j-1];
18. }
19. printf("%lld\n",dp[n][k]);
20. }

26. [一元三次方程求解](http://blog.csdn.net/buctyyzyn/article/details/43873003)

问题描述

　　有形如：ax3+bx2+cx+d=0 这样的一个一元三次方程。给出该方程中各项的系数(a，b，c，d 均为实数)，并约定该方程存在三个不同实根(根的范围在-100至100之间)，且根与根之差的绝对值>=1。要求三个实根。。

输入格式

　　四个实数：a，b，c，d

输出格式

　　由小到大依次在同一行输出这三个实根(根与根之间留有空格)，并精确到小数点后2位

样例输入

1 -5 -4 20

样例输出

-2.00 2.00 5.00

解题思路：

根的范围在-100--100之间，且要精确到后俩位，我们可以把根的范围过大100倍，x1=(x-0.05)/100,x2=(x+0.05)/100,根据根的特点可以确定f(x1)\*f(x2)<0,因此遇到这种情况即是满足情况的解。

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43873003)

1. #include<cstdio>
2. #include<cstring>
3. **float** a,b,c,d;
4. **float** f(**float** x)
5. {
6. **return** x\*x\*x+b\*1.00/a\*x\*x+c\*1.00/a\*x+d\*1.00/a;
7. }
8. **int** main()
9. {
10. scanf("%f%f%f%f",&a,&b,&c,&d);
11. **for**(**int** x=-10000;x<=10000;x++)
12. {
13. **float** x1=(x-0.05)\*1.00/100,x2=(x+0.05)\*1.00/100;
14. **if**(f(x1)\*f(x2)<0)
15. printf("%.2f ",x\*1.00/100);
16. }
17. printf("\n");
18. }

26.  [**Car的旅行路线**](http://blog.csdn.net/buctyyzyn/article/details/43887629)

问题描述

　　又到暑假了，住在城市A的Car想和朋友一起去城市B旅游。她知道每个城市都有四个飞机场，分别位于一个矩形的四个顶点上，同一个城市中两个机场之间有一 条笔直的高速铁路，第I个城市中高速铁路了的单位里程价格为Ti，任意两个不同城市的机场之间均有航线，所有航线单位里程的价格均为t。  
　　那么Car应如何安排到城市B的路线才能尽可能的节省花费呢?她发现这并不是一个简单的问题，于是她来向你请教。  
　　找出一条从城市A到B的旅游路线，出发和到达城市中的机场可以任意选取，要求总的花费最少。

输入格式

　　的第一行有四个正整数s，t，A，B。  
　　S表示城市的个数，t表示飞机单位里程的价格，A，B分别为城市A，B的序号，(1<=A，B<=S)。

　　接下来有S行，其中第I行均有7个正整数xi1，yi1，xi2，yi2，xi3，yi3，Ti，这当中的(xi1，yi1)，(xi2，yi2)，(xi3，yi3)分别是第I个城市中任意三个机场的坐标，T I为第I个城市高速铁路单位里程的价格。

解题思路：

首先最多400个点的图，用哪种求最短路的方法都可以，我这里选用了spfa；

1：给出了每一个城市的三个机场坐标，需要求出最后一个坐标，方法如下

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43887629)

1. **void** find(node &a)
2. {
3. **if**(((a.x[1]-a.x[0])\*(a.x[2]-a.x[0])+(a.y[1]-a.y[0])\*(a.y[2]-a.y[0]))==0)
4. {
5. a.x[3]=a.x[1]+a.x[2]-a.x[0];
6. a.y[3]=a.y[1]+a.y[2]-a.y[0];
7. }
8. **else** **if**(((a.x[0]-a.x[1])\*(a.x[2]-a.x[1])+(a.y[0]-a.y[1])\*(a.y[2]-a.y[1]))==0)
9. {
10. a.x[3]=a.x[0]+a.x[2]-a.x[1];
11. a.y[3]=a.y[0]+a.y[2]-a.y[1];
12. }
13. **else**
14. {
15. a.x[3]=a.x[0]+a.x[1]-a.x[2];
16. a.y[3]=a.y[0]+a.y[1]-a.y[2];
17. }
18. }

2：构图，站点内部不同机场连边，不同城市的机场连边；

3：可以弄一个起始点连接A，路程为0；一个结束点连接B，路程为0，这样就可以方便处理最短路。

代码如下：

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43887629)

1. #include<cstdio>
2. #include<cstring>
3. #include<cmath>
4. #include<algorithm>
5. #include<queue>
6. **using** **namespace** std;
7. #define maxn 200000
8. #define inf 0x3f3f3f3f
9. **struct** node
10. {
11. **double** x[4],y[4];
12. **int** note[4];
13. **double** w;
14. };
15. **double** dis(node a,**int** u,node b,**int** v)
16. {
17. **return** sqrt((a.x[u]-b.x[v])\*(a.x[u]-b.x[v])+(a.y[u]-b.y[v])\*(a.y[u]-b.y[v]));
18. }
19. **struct** Edge
20. {
21. **int** v;
22. **double** w;
23. **int** next;
24. }edge[3\*maxn];
25. **int** head[maxn],vis[maxn],cnt,n,Max,Min;
26. **double** d[maxn];
27. **int** cnn;
28. **int** s,tt,a,b;
29. **void** addedge(**int** u,**int** v,**double** w)
30. {
31. edge[cnt].v=v;
32. edge[cnt].w=w;
33. edge[cnt].next=head[u];
34. head[u]=cnt++;
35. }
36. **void** spfa()
37. {
38. **for**(**int** i=Min;i<=Max;i++) d[i]=inf;
39. d[Min]=0;
40. queue<**int** >q;
41. **while**(!q.empty()) q.pop();
42. q.push(Min);
43. **while**(!q.empty())
44. {
45. **int** x=q.front();
46. vis[x]=0;
47. q.pop();
48. **for**(**int** e=head[x];e!=-1;e=edge[e].next)
49. **if**(d[edge[e].v]>d[x]+edge[e].w)
50. {
51. d[edge[e].v]=d[x]+edge[e].w;
52. **if**(!vis[edge[e].v])
53. {
54. q.push(edge[e].v);
55. vis[edge[e].v]=1;
56. }
57. }
58. }
59. }
60. **void** find(node &a)
61. {
62. **if**(((a.x[1]-a.x[0])\*(a.x[2]-a.x[0])+(a.y[1]-a.y[0])\*(a.y[2]-a.y[0]))==0)
63. {
64. a.x[3]=a.x[1]+a.x[2]-a.x[0];
65. a.y[3]=a.y[1]+a.y[2]-a.y[0];
66. }
67. **else** **if**(((a.x[0]-a.x[1])\*(a.x[2]-a.x[1])+(a.y[0]-a.y[1])\*(a.y[2]-a.y[1]))==0)
68. {
69. a.x[3]=a.x[0]+a.x[2]-a.x[1];
70. a.y[3]=a.y[0]+a.y[2]-a.y[1];
71. }
72. **else**
73. {
74. a.x[3]=a.x[0]+a.x[1]-a.x[2];
75. a.y[3]=a.y[0]+a.y[1]-a.y[2];
76. }
77. }
78. **void** init()
79. {
80. memset(vis,0,**sizeof**(vis));
81. memset(head,-1,**sizeof**(head));
82. memset(edge,0,**sizeof**(edge));
83. cnt=0;
84. Min=0;
85. cnn=1;
86. }
87. **void** addin(node a)
88. {
89. **for**(**int** i=0;i<4;i++)
90. {
91. **for**(**int** j=0;j<4;j++)
92. {
93. **if**(i!=j)
94. addedge(a.note[i],a.note[j],a.w\*dis(a,i,a,j));
95. addedge(a.note[j],a.note[i],a.w\*dis(a,j,a,i));
96. }
97. }
98. }
99. **void** addout(node a,node b)
100. {
101. **for**(**int** i=0;i<4;i++)
102. {
103. **for**(**int** j=0;j<4;j++)
104. {
105. addedge(a.note[i],b.note[j],tt\*dis(a,i,b,j));
106. addedge(b.note[j],a.note[i],tt\*dis(a,i,b,j));
107. }
108. }
109. }
110. **void** addbegin(node a)
111. {
112. **for**(**int** i=0;i<4;i++)
113. {
114. addedge(0,a.note[i],0);
115. addedge(a.note[i],0,0);
116. }
117. }
118. **void** addend(node a)
119. {
120. **for**(**int** i=0;i<4;i++)
121. {
122. addedge(cnn,a.note[i],0);
123. addedge(a.note[i],cnn,0);
124. }
125. }
126. **int** main()
127. {
128. init();
129. scanf("%d%d%d%d",&s,&tt,&a,&b);
130. node point[110];
132. **for**(**int** i=1;i<=s;i++)
133. {
134. scanf("%lf%lf%lf%lf%lf%lf%lf",&point[i].x[0],&point[i].y[0],&point[i].x[1],&point[i].y[1],&point[i].x[2],&point[i].y[2],&point[i].w);
135. find(point[i]);
136. **for**(**int** j=0;j<4;j++)
137. point[i].note[j]=cnn++;
138. addin(point[i]);
139. }
140. Max=cnn;
141. **for**(**int** i=1;i<=s;i++)
142. {
143. **for**(**int** j=1;j<=s;j++)
144. {
145. **if**(i!=j)
146. {
147. addout(point[i],point[j]);
148. }
149. }
150. }
151. addbegin(point[a]);
152. addend(point[b]);
153. spfa();
154. printf("%.1lf",d[cnn]);
155. }

27. [麦森数](http://blog.csdn.net/buctyyzyn/article/details/43902549)

问题描述

　　形如2P-1的素数称为麦森数，这时P一定也是个素数。但反过来不一定，即如果P是个素数，2P-1不一定也是素数。到1998年底，人们已找到了37个麦森数。最大的一个是P=3021377，它有909526位。麦森数有许多重要应用，它与完全数密切相关。  
　　任务：从文件中输入P（1000<P<3100000），计算2P-1的位数和最后500位数字（用十进制高精度数表示）

输入格式

　　文件中只包含一个整数P（1000<P<3100000）

输出格式

　　第一行：十进制高精度数2P-1的位数。  
　　第2-11行：十进制高精度数2P-1的最后500位数字。（每行输出50位，共输出10行，不足500位时高位补0）  
　　不必验证2P-1与P是否为素数。

样例输入

1279

样例输出

386  
00000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000  
00000000000000104079321946643990819252403273640855  
38615262247266704805319112350403608059673360298012  
23944173232418484242161395428100779138356624832346  
49081399066056773207629241295093892203457731833496  
61583550472959420547689811211693677147548478866962  
50138443826029173234888531116082853841658502825560  
46662248318909188018470682222031405210266984354887

32958028878050869736186900714720710555703168729087

解题思路：

关于位数，对2^p取10的对数即可，虽说最后减了1，但是不会影响最后的位数，最终为p\*log10(2)+1;

因为数值比较大，可以用一个数组存储，一位表示5位，这样100即可；

普通的乘法肯定要超时，我们可以借助位运算，一次左移10次，最终对于不足10次左移的再一位一位移即可；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43902549)

1. #include<cstdio>
2. #include<math.h>
3. #include<cstring>
4. #define max 100000
5. **int** main()
6. {
7. **int** p;
8. scanf("%d",&p);
9. printf("%d\n",(**int**)(p\*log10(2))+1);
10. **int** ans[110];
11. memset(ans,0,**sizeof**(ans));
12. ans[0]=1;
13. **int** left=p%10;
14. p/=10;
15. **for**(**int** i=1;i<=p;i++)
16. {
17. **for**(**int** j=0;j<=100;j++)
18. {
19. ans[j]<<=10;
20. }
21. **for**(**int** j=0;j<=100;j++)
22. {
23. **if**(ans[j]>=max)
24. {
25. ans[j+1]+=ans[j]/max;
26. ans[j]%=max;
27. }
28. }
29. }
30. **for**(**int** i=1;i<=left;i++)
31. {
32. **for**(**int** j=0;j<=100;j++)
33. ans[j]<<=1;
34. **for**(**int** j=0;j<=100;j++)
35. {
36. **if**(ans[j]>=max)
37. {
38. ans[j+1]+=ans[j]/max;
39. ans[j]%=max;
40. }
41. }
42. }
43. ans[0]-=1;
44. **for**(**int** i=1;i<100;i++)
45. {
46. **if**(ans[i-1]<0)
47. {
48. ans[i]-=1;
49. ans[i-1]+=max;
50. }
51. **else**
52. **break**;
53. }
54. **for**(**int** i=99;i>=0;i--)
55. {
56. printf("%05d",ans[i]);
57. **if**((100-i)%10==0)
58. printf("\n");
59. }
60. }
61. [**FBI树**](http://blog.csdn.net/buctyyzyn/article/details/43914185)

问题描述

　　我们可以把由“0”和“1”组成的字符串分为三类：全“0”串称为B串，全“1”串称为I串，既含“0”又含“1”的串则称为F串。  
　　FBI树是一种二叉树，它的结点类型也包括F结点，B结点和I结点三种。由一个长度为2N的“01”串S可以构造出一棵FBI树T，递归的构造方法如下：  
　　1)T的根结点为R，其类型与串S的类型相同；  
　　2)若串S的长度大于1，将串S从中间分开，分为等长的左右子串S1和S2；由左子串S1构造R的左子树T1，由右子串S2构造R的右子树T2。  
　　现在给定一个长度为2N的“01”串，请用上述构造方法构造出一棵FBI树，并输出它的后序遍历序列。

输入格式

　　第一行是一个整数N（0 <= N <= 10），第二行是一个长度为2N的“01”串。

输出格式

　　包括一行，这一行只包含一个字符串，即FBI树的后序遍历序列。

样例输入

3  
10001011

样例输出

IBFBBBFIBFIIIFF

解题思路：

直接递归建树，通过判断左右孩子是否相同来判断是‘F’,'B','I'中的哪一个；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43914185)

1. #include<cstdio>
2. #include<iostream>
3. #include<string>
4. #include<cstring>
5. #include<algorithm>
6. **using** **namespace** std;
7. **struct** node{
8. **char** ch;
9. **struct** node \*lc,\*rc;
10. };
11. **void** create(node \*&T,string s)
12. {
14. **int** len=s.length();
15. **if**(len==1)
16. {
17. T=(node\*)malloc(**sizeof**(node));
18. **if**(s[0]=='1')
19. T->ch='I';
20. **else**
21. T->ch='B';
22. T->lc=T->rc=NULL;
23. **return** ;
24. }
25. **else**
26. {
27. T=(node\*)malloc(**sizeof**(node));
28. create(T->lc,s.substr(0,len/2));
29. create(T->rc,s.substr(len/2,len/2));
31. **if**(T->lc->ch==T->rc->ch)
32. {
33. T->ch=T->lc->ch;
34. }
35. **else**
36. {
37. T->ch='F';
38. }
39. }
40. }
41. **void** dfs(node \*T)
42. {
43. **if**(T!=NULL)
44. {
45. dfs(T->lc);
46. dfs(T->rc);
47. cout<<T->ch;
48. }
49. }
50. **int** main()
51. {
52. **int** n;
53. **int** ans=1;
54. string ss;
55. cin>>n;
56. cin>>ss;
57. ans<<=n;
58. node \*T;
59. create(T,ss);
60. dfs(T);
61. cout<<endl;
62. }
63. [校门外的树](http://blog.csdn.net/buctyyzyn/article/details/43927627)

问题描述

　　某校大门外长度为L的马路上有一排树，每两棵相邻的树之间的间隔都是1米。我们可以把马路看成一个数轴，马路的一端在数轴0的位置，另一端在L的位置；数 轴上的每个整数点，即0，1，2，……，L，都种有一棵树。  
　　由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已 知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把这些区域中的树（包括区域端点处的两棵树）移走。你的任务是计算将这些树 都移走后，马路上还有多少棵树。

输入格式

　　输入文件的第一行有两个整数L（1 <= L <= 10000）和 M（1 <= M <= 100），L代表马路的长度，M代表区域的数目，L和M之间用一个空格隔开。接下来的M行每行包含两个不同的整数，用一个空格隔开，表示一个区域的起始点 和终止点的坐标。

输出格式

　　输出文件包括一行，这一行只包含一个整数，表示马路上剩余的树的数目。

样例输入

500 3  
150 300  
100 200  
470 471

样例输出

298

解题思路：

线段树区间处理问题，顺便自己写了一个线段树当做练手了；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43927627)

1. #include<cstdio>
2. #include<cstring>
3. #include<algorithm>
4. #include<string>
5. **using** **namespace** std;
6. #define maxn 100000
7. #define L(a) (a)<<1
8. #define R(a) (a)<<1|1
9. **struct** Node{
10. **int** sum;
11. **int** add;
12. **int** set;
13. **int** left,right;
14. }node[3\*maxn];
15. **int** ans;
16. **int** a,b;
17. **void** create(**int** num,**int** l,**int** r)
18. {
19. node[num].left=l;
20. node[num].right=r;
21. **if**(l==r)
22. {
23. node[num].sum=1;
24. node[num].set=-1;
25. node[num].add=0;
26. **return** ;
27. }
28. **int** mid=(l+r)>>1;
29. create(L(num),l,mid);
30. create(R(num),mid+1,r);
31. node[num].sum=node[L(num)].sum+node[R(num)].sum;
32. node[num].set=-1;
33. node[num].add=0;
35. }
37. **void** maintain(**int** num,**int** l,**int** r)
38. {
39. node[num].sum=1;
40. **if**(r>l)
41. {
42. node[num].sum=node[L(num)].sum+node[R(num)].sum;
43. }
45. }
46. **void** pushup(**int** num)
47. {
48. node[num].sum=node[L(num)].sum+node[R(num)].sum;
49. }
50. **void** pushdown(**int** num)
51. {
52. **if**(node[num].set==0)
53. {
54. node[L(num)].set=node[R(num)].set=node[num].set;
55. node[L(num)].sum=0;
56. node[R(num)].sum=0;
57. node[num].set=-1;
58. }
59. }
60. **void** updata(**int** num,**int** l,**int** r,**int** v)
61. {
62. **if**(node[num].left==l&&node[num].right==r)
63. {
64. node[num].set=v;
65. node[num].sum=(r-l+1)\*v;
66. **return** ;
67. }
68. **else**
69. {
70. pushdown(num);
71. **int** mid=(node[num].left+node[num].right)>>1;
72. **if**(r<=mid)
73. updata(L(num),l,r,v);
74. **else** **if**(l>mid)
75. updata(R(num),l,r,v);
76. **else**
77. {
78. updata(L(num),l,mid,v);
79. updata(R(num),mid+1,r,v);
80. }
81. pushup(num);
82. }
83. }
84. **void** query(**int** num,**int** l,**int** r)
85. {
86. **if**(node[num].left==l&&node[num].right==r)
87. {
88. ans+=node[num].sum;
89. **return** ;
90. }
91. **int** mid=(node[num].left+node[num].right)>>1;
92. **if**(r<=mid)
93. query(L(num),l,r);
94. **else** **if**(l>mid)
95. query(R(num),l,r);
96. **else**
97. {
98. query(L(num),l,mid);
99. query(R(num),mid+1,r);
100. }
102. }
103. **int** main()
104. {
105. **int** n,m;
106. scanf("%d%d",&n,&m);
107. create(1,0,n);
108. ans=0;
109. **for**(**int** i=0;i<m;i++)
110. {
111. scanf("%d%d",&a,&b);
112. updata(1,a,b,0);
113. }
114. query(1,0,n);
115. printf("%d\n",ans);
116. }

31.  [**JAM计数法**](http://blog.csdn.net/buctyyzyn/article/details/43931349)

问题描述

　　Jam是个喜欢标新立异的科学怪人。他不使用阿拉伯数字计数，而是使用小写英文字母计数，他觉得这样做，会使世界更加丰富多彩。在他的计数法中，每个数字的位数都是相同的（使用相同个数的字母），英文字母按原先的顺序，排在前面的字母小于排在它后面的字母。我们把这样的“数字”称为Jam数字。在Jam数字中，每个字母互不相同，而且从左到右是严格递增的。每次，Jam还指定使用字母的范围，例如，从2到10，表示只能使用{b,c,d,e,f,g,h,i,j}这些字母。如果再规定位数为5，那么，紧接在Jam数字“bdfij”之后的数字应该是“bdghi”。（如果我们用U、V依次表示Jam数字“bdfij”与“bdghi”，则U<V< span>，且不存在Jam数字P，使U<P<V< span>）。你的任务是：对于从文件读入的一个Jam数字，按顺序输出紧接在后面的5个Jam数字，如果后面没有那么多Jam数字，那么有几个就输出几个。

输入格式

　　有2行，第1行为3个正整数，用一个空格隔开：  
　　s t w  
　　（其中s为所使用的最小的字母的序号，t为所使用的最大的字母的序号。w为数字的位数，这3个数满足：1≤s<T≤26, 2≤w≤t-s ）  
　　第2行为具有w个小写字母的字符串，为一个符合要求的Jam数字。  
　　所给的数据都是正确的，不必验证。

输出格式

　　最多为5行，为紧接在输入的Jam数字后面的5个Jam数字，如果后面没有那么多Jam数字，那么有几个就输出几个。每行只输出一个Jam数字，是由w个小写字母组成的字符串，不要有多余的空格。

样例输入

2 10 5  
bdfij

样例输出

bdghi  
bdghj  
bdgij  
bdhij

befgh

**解题思路：从字符串的最后一位往前找满足条件的字符**

**1：在s,t范围内该字符后面的字符个数大于等于该字符在字符串中的位置的后面的字符个数**

**如果找到这样的字符的话，顺序的替换字符串中从该字符开始的字符，按递增序；**

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43931349)

1. #include<cstdio>
2. #include<cstring>
3. **int** s,t,w;
4. **int** cnt;
5. **void** dfs(**char** \*s)
6. {
7. **if**(cnt==5)
8. **return** ;
9. **for**(**int** i=w-1;i>=0;i--)
10. {
11. **if**(cnt==5)
12. **return** ;
13. **int** tt=s[i]-'a'+1;
14. **if**(t-tt>=w-i)
15. {
16. **int** k=i;
17. **for**(**int** j=tt;j<tt+w-i;j++)
18. {
19. s[k++]='a'+j;
20. }
21. cnt++;
22. printf("%s\n",s);
23. dfs(s);
24. }
25. }
26. }
27. **int** main()
28. {
29. scanf("%d%d%d",&s,&t,&w);
30. **char** ss[100];
31. cnt=0;
32. scanf("%s",ss);
33. dfs(ss);
34. }

32. [数列](http://blog.csdn.net/buctyyzyn/article/details/43931949)

问题描述

　　给定一个正整数k(3≤k≤15),把所有k的方幂及所有有限个互不相等的k的方幂之和构成一个递增的序列，例如，当k=3时，这个序列是：  
　　1，3，4，9，10，12，13，…  
　　（该序列实际上就是：30，31，30+31，32，30+32，31+32，30+31+32，…）  
　　请你求出这个序列的第N项的值（用10进制数表示）。  
　　例如，对于k=3，N=100，正确答案应该是981。

输入格式

　　只有1行，为2个正整数，用一个空格隔开：  
　　k N  
　　（k、N的含义与上述的问题描述一致，且3≤k≤15，10≤N≤1000）。

输出格式

　　计算结果，是一个正整数（在所有的测试数据中，结果均不超过2.1\*109）。（整数前不要有空格和其他符号）。

样例输入

3 100

样例输出

981

**解题思路：**

**由于第N个数字是互不相等的k的方幂之和,所以每个方幂在这个数字里面最多出现一次,每个方幂是否出现可以用一个bit位来表示,幂次代表bit位的位置,于是k^3表示成1000,k^0表示成0001,所有的这类数字按从大到小的顺序排出来就是0001,0010,0011,0100……第N项其实就是N转换成二进制之后,再将相应的位转换成相应的k的方幂!**

**唯一郁闷的是蓝桥杯的OJ不知道为什么用pow函数会报错**

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43931949)

1. #include<cstdio>
2. #include<algorithm>
3. #include<string>
4. #include<cmath>
5. **using** **namespace** std;
6. **long** **long** p(**int** k,**int** i)
7. {
8. **long** **long** ans=1;
9. **if**(i==0)
10. **return** 1;
11. **for**(**int** j=1;j<=i;j++)
12. ans\*=k;
13. **return** ans;
14. }
15. **int** main()
16. {
17. **int** k,n;
18. scanf("%d%d",&k,&n);
19. **int** s[11];
20. **int** t=0;
21. **while**(n)
22. {
23. s[t++]=n&1;
24. n>>=1;
25. }
26. **long** **long** ans=0;
27. **for**(**int** i=t-1;i>=0;i--)
28. {
29. **if**(s[i])
30. ans+=s[i]\*p(k,i);
31. }
32. printf("%lld",ans);
33. }
34. [纪念品分组](http://blog.csdn.net/buctyyzyn/article/details/43932103)

问题描述

　　元旦快到了，校学生会让乐乐负责新年晚会的纪念品发放工作。为使得参加晚会的同学所获得的纪念品价值 相对均衡，他要把购来的纪念品根据价格进行分组，但每组最多只能包括两件纪念品，并且每组纪念品的价格之和不能超过一个给定的整数。为了保证在尽量短的时 间内发完所有纪念品，乐乐希望分组的数目最少。  
　　你的任务是写一个程序，找出所有分组方案中分组数最少的一种，输出最少的分组数目。

输入格式

　　输入包含n+2行：  
　　第1行包括一个整数w，为每组纪念品价格之和的上限。  
　　第2行为一个整数n，表示购来的纪念品的总件数。  
　　第3~n+2行每行包含一个正整数pi (5 <= pi <= w)，表示所对应纪念品的价格。

输出格式

　　输出仅一行，包含一个整数，即最少的分组数目。

样例输入

100  
9  
90  
20  
20  
30  
50  
60  
70  
80  
90

样例输出

6

数据规模和约定

　　50%的数据满足：1 <= n <= 15  
　　100%的数据满足：1 <= n <= 30000, 80 <= w <= 200

**解题思路：**

  每组最多俩个物品，为了让分组最少，则每组中尽可能放多的物品，因此，对数组从小到大排序一次，用s,e控制将要取得数字位置，先取e，如果再取s不超过W，则取，反之，不取；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43932103)

1. #include<cstdio>
2. #include<algorithm>
3. **using** **namespace** std;
4. **int** main()
5. {
6. **int** w,n;
7. **int** num[30010];
8. scanf("%d%d",&w,&n);
9. **for**(**int** i=0;i<n;i++)
10. scanf("%d",&num[i]);
11. sort(num,num+n);
12. **int** s=0,e=n-1;
13. **int** ans=0;
14. **while**(s<=e)
15. {
16. **if**(num[s]+num[e]<=w)
17. {
18. s++;
19. e--;
20. ans++;
21. }
22. **else**
23. {
24. e--;
25. ans++;
26. }
27. }
28. printf("%d\n",ans);
29. }

35. [传球游戏](http://blog.csdn.net/buctyyzyn/article/details/43937697)

问题描述

　　上体育课的时候，小蛮的老师经常带着同学们一起做游戏。这次，老师带着同学们一起做传球游戏。  
　　游戏规则是这样的：n个同学站成一个圆圈，其中的一个同学手里拿着一个球，当老师吹哨子时开始传球，每个同学可以把球传给自己左右的两个同学中的一个（左右任意），当老师再次吹哨子时，传球停止，此时，拿着球没传出去的那个同学就是败者，要给大家表演一个节目。  
　　聪明的小蛮提出一个有趣的问题：有多少种不同的传球方法可以使得从小蛮手里开始传的球，传了m次以后，又回到小蛮手里。两种传球的方法被视作不同的方法，当且仅当这两种方法中，接到球的同学按接球顺序组成的序列是不同的。比如有3个同学1号、2号、3号，并假设小蛮为1号，球传了3次回到小蛮手里的方式有1->2->3->1和1->3->2->1，共2种。

输入格式

　　共一行，有两个用空格隔开的整数n，m（3<=n<=30，1<=m<=30）。

输出格式

　　t共一行，有一个整数，表示符合题意的方法数。

样例输入

3 3

样例输出

2

数据规模和约定

　　40%的数据满足：3<=n<=30，1<=m<=20  
　　100%的数据满足：3<=n<=30，1<=m<=30

# 解题思路：

dp[i][j]表示传i次球，将球传到j手里的的方法数；

 传一次球可以传到n，可以传到2，所以dp[1][n]=1;dp[1][2]=1;

 传0次球，则在第一个人手中，dp[o][1]=1;

状态转移方程为

dp[i][j]=dp[i-1][j-1]+dp[i-1][j+1];

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43937697)

1. #include<cstdio>
2. #include<cstring>
3. #include<algorithm>
4. **using** **namespace** std;
5. **int** main()
6. {
7. **int** dp[35][35];
8. **int** n,m;
9. scanf("%d%d",&n,&m);
10. memset(dp,0,**sizeof**(dp));
11. dp[0][1]=1;
12. dp[1][n]=1;
13. dp[1][2]=1;
14. **for**(**int** i=1;i<=m;i++)
15. {
16. **for**(**int** j=1;j<=n;j++)
17. {
18. **if**(j==1)
19. dp[i][j]=dp[i-1][n]+dp[i-1][2];
20. **else** **if**(j==n)
21. dp[i][j]=dp[i-1][1]+dp[i-1][n-1];
22. **else**
23. dp[i][j]=dp[i-1][j-1]+dp[i-1][j+1];
24. }
25. }
26. printf("%d\n",dp[m][1]);
27. }

# 36.[传纸条](http://blog.csdn.net/buctyyzyn/article/details/43953401)

问题描述

　　小渊和小轩是好朋友也是同班同学，他们在一起总有谈不完的话题。一次素质拓展活动中，班上同学安排做成一个m行n列的矩阵，而小渊和小轩被安排在矩阵对角线的两端，因此，他们就无法直接交谈了。幸运的是，他们可以通过传纸条来进行交流。纸条要经由许多同学传到对方手里，小渊坐在矩阵的左上角，坐标(1,1)，小轩坐在矩阵的右下角，坐标(m,n)。从小渊传到小轩的纸条只可以向下或者向右传递，从小轩传给小渊的纸条只可以向上或者向左传递。  
　　在活动进行中，小渊希望给小轩传递一张纸条，同时希望小轩给他回复。班里每个同学都可以帮他们传递，但只会帮他们一次，也就是说如果此人在小渊递给小轩纸条的时候帮忙，那么在小轩递给小渊的时候就不会再帮忙。反之亦然。  
　　还有一件事情需要注意，全班每个同学愿意帮忙的好感度有高有低（注意：小渊和小轩的好心程度没有定义，输入时用0表示），可以用一个0-100的自然数来表示，数越大表示越好心。小渊和小轩希望尽可能找好心程度高的同学来帮忙传纸条，即找到来回两条传递路径，使得这两条路径上同学的好心程度只和最大。现在，请你帮助小渊和小轩找到这样的两条路径。

输入格式

　　输入第一行有2个用空格隔开的整数m和n，表示班里有m行n列（1<=m,n<=50）。  
　　接下来的m行是一个m\*n的矩阵，矩阵中第i行j列的整数表示坐在第i行j列的学生的好心程度。每行的n个整数之间用空格隔开。

输出格式

　　输出一行，包含一个整数，表示来回两条路上参与传递纸条的学生的好心程度之和的最大值。

样例输入

3 3  
0 3 9  
2 8 5  
5 7 0

样例输出

34

数据规模和约定

　　30%的数据满足：1<=m,n<=10

　　100%的数据满足：1<=m,n<=50

# 解题思路：

这道题和方格取数极为相似，但要注意判重；

dp[i][j][k][l]:（i，j）第一个纸条的位置，（k,l）表示第二个纸条的位置；

首先你得保证（i,j）(k,l)不在同一个位置，但是（m,n）(m,n)要进行特判，否则没有答案；

其次，也要保证可能转移到现在的位置的位置也不能相同；

dp[i][j][k][l]的结果就是可能的四种方案中的最大值+对应的俩个格子里的好感度；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43953401)

1. #include<cstdio>
2. #include<algorithm>
3. **using** **namespace** std;
4. **int** dp[51][51][51][51]={0};
5. **int** main()
6. {
7. **int** num[51][51]={0};
8. **int** m,n;
9. scanf("%d%d",&m,&n);
10. **for**(**int** i=1;i<=m;i++)
11. **for**(**int** j=1;j<=n;j++)
12. scanf("%d",&num[i][j]);
13. **for**(**int** i=1;i<=m;i++)
14. {
15. **for**(**int** j=1;j<=n;j++)
16. {
17. **for**(**int** k=1;k<=m;k++)
18. {
19. **for**(**int** l=1;l<=n;l++)
20. {
21. **if**((i==k&&j==l)&&(i!=m&&j!=n))
22. **continue**;
23. **int** tem=0;
24. **if**(i-1!=k-1&&j!=l)
25. tem=max(tem,dp[i-1][j][k-1][l]);
26. **if**(i-1!=k&&j!=l-1)
27. tem=max(tem,dp[i-1][j][k][l-1]);
28. **if**(i!=k-1&&j-1!=l)
29. tem=max(tem,dp[i][j-1][k-1][l]);
30. **if**(i!=k&&j-1!=l-1)
31. tem=max(tem,dp[i][j-1][k][l-1]);
32. dp[i][j][k][l]=max(dp[i][j][k][l],tem+num[i][j]+num[k][l]);
33. }
34. }
35. }
36. }
37. printf("%d\n",dp[m][n][m][n]);
38. }

## 38.[接水问题](http://blog.csdn.net/buctyyzyn/article/details/43954115)

问题描述

　　学校里有一个水房，水房里一共装有m 个龙头可供同学们打开水，每个龙头每秒钟的 供水量相等，均为1。 现在有n 名同学准备接水，他们的初始接水顺序已经确定。将这些同学按接水顺序从1 到n 编号，i 号同学的接水量为wi。接水开始时，1 到m 号同学各占一个水龙头，并同时打 开水龙头接水。当其中某名同学j 完成其接水量要求wj 后，下一名排队等候接水的同学k 马上接替j 同学的位置开始接水。这个换人的过程是瞬间完成的，且没有任何水的浪费。即 j 同学第x 秒结束时完成接水，则k 同学第x+1 秒立刻开始接水。若当前接水人数n’不足m， 则只有n’个龙头供水，其它m−n’个龙头关闭。 现在给出n 名同学的接水量，按照上述接水规则，问所有同学都接完水需要多少秒。

输入格式

　　第1 行2 个整数n 和m，用一个空格隔开，分别表示接水人数和龙头个数。 第2 行n 个整数w1、w2、……、wn，每两个整数之间用一个空格隔开，wi 表示i 号同 学的接水量。

输出格式

　　输出只有一行，1 个整数，表示接水所需的总时间。

样例输入

5 3  
4 4 1 2 1

样例输出

4

样例输入

8 4  
23 71 87 32 70 93 80 76

样例输出

163

输入输出样例 1 说明

　　第1 秒，3 人接水。第1 秒结束时，1、2、3 号同学每人的已接水量为1，3 号同学接完  
　　水，4 号同学接替3 号同学开始接水。  
　　第2 秒，3 人接水。第2 秒结束时，1、2 号同学每人的已接水量为2，4 号同学的已接  
　　水量为1。  
　　第3 秒，3 人接水。第3 秒结束时，1、2 号同学每人的已接水量为3，4 号同学的已接  
　　水量为2。4 号同学接完水，5 号同学接替4 号同学开始接水。  
　　第4 秒，3 人接水。第4 秒结束时，1、2 号同学每人的已接水量为4，5 号同学的已接  
　　水量为1。1、2、5 号同学接完水，即所有人完成接水。  
　　总接水时间为4 秒。

数据规模和约定

　　1 ≤ n ≤ 10000，1 ≤m≤ 100 且m≤ n；

　　1 ≤ wi ≤ 100。

# 解题思路：

这题要求最先接完水的水龙头让紧接着的没有接水的同学替换，最后输出所有人接完水需要的时间。

我们可以建立以优先队列，从小到大排序，每次取出队首，再将队首+紧接着同学的接水量入队，答案即是队尾元素；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43954115)

1. #include<cstdio>
2. #include<algorithm>
3. #include<queue>
4. **using** **namespace** std;
5. **int** main()
6. {
7. **int** n,m;
8. **int** num[10010];
9. scanf("%d%d",&n,&m);
10. **for**(**int** i=0;i<n;i++)
11. scanf("%d",&num[i]);
12. priority\_queue<**int**,vector<**int**>,greater<**int**> > q;
13. **while**(!q.empty()) q.pop();
14. **for**(**int** i=0;i<m;i++)
15. q.push(num[i]);
16. **while**(m<n)
17. {
18. **int** t=q.top();
19. q.pop();
20. q.push(t+num[m]);
21. m++;
22. }
23. **while**(q.size()>1)
24. q.pop();
25. **int** ans=q.top();
26. q.pop();
27. printf("%d\n",ans);
28. }
29. [日期计算](http://blog.csdn.net/buctyyzyn/article/details/43987911)

问题描述

　　已知2011年11月11日是星期五，问YYYY年MM月DD日是星期几？注意考虑闰年的情况。尤其是逢百年不闰，逢400年闰的情况。

输入格式

　　输入只有一行  
　　YYYY MM DD

输出格式

　　输出只有一行  
　　W

数据规模和约定

　　1599 <= YYYY <= 2999  
　　1 <= MM <= 12  
　　1 <= DD <= 31，且确保测试样例中YYYY年MM月DD日是一个合理日期  
　　1 <= W <= 7，分别代表周一到周日

样例输入

2011 11 11

样例输出

5

# 解题思路：

这道题思路上没有什么难点，就是简单的计算给的日期距离2011,11，11相差几天，一些细节的处理需要注意；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43987911)[在CODE上查看代码片](https://code.csdn.net/snippets/609350)

1. #include<cstdio>
2. **int** num1[12]={31,28,31,30,31,30,31,31,30,31,30,31};
3. **int** num2[12]={31,29,31,30,31,30,31,31,30,31,30,31};
4. **int** change[7]={5,6,7,1,2,3,4};
5. **int** change1[7]={5,4,3,2,1,7,6};
6. **bool** judge(**int** a,**int** b,**int** c)
7. {
8. **if**(a<2011)
9. **return** **true**;
10. **else** **if**(a==2011)
11. {
12. **if**(b<11)
13. **return** **true**;
14. **else** **if**(b==11)
15. {
16. **if**(c<11)
17. **return** **true**;
18. **else**
19. **return** **false**;
20. }
21. **else**
22. **return** **false**;
23. }
24. **else**
25. **return** **false**;
26. }
27. **bool** r(**int** i)
28. {
29. **if**((i%4==0&&i%100!=0)||i%400==0)
30. **return** **true**;
31. **return** **false**;
32. }
33. **int** main()
34. {
35. **int** year,month,day;
36. scanf("%d%d%d",&year,&month,&day);
37. **if**(year==2011&&month==11&&day==11)
38. printf("5\n");
39. **else**
40. {
41. **if**(judge(year,month,day))
42. {
43. **int** ans=0;
44. **for**(**int** i=year+1;i<2011;i++)
45. {
46. **if**(r(i))
47. ans+=1;
48. }
49. ans+=(2011-year)\*365;
50. ans-=51;
51. **if**(r(year))
52. {
53. ans+=366;
54. **for**(**int** i=0;i<month-1;i++)
55. ans-=num2[i];
56. ans-=day;
57. ans+=1;
58. }
59. **else**
60. {
61. ans+=365;
62. **for**(**int** i=0;i<month-1;i++)
63. ans-=num1[i];
64. ans-=day;
65. ans+=1;
66. }
67. printf("%d\n",change1[ans%7]);
68. }
69. **else**
70. {
71. **int** ans=0;
72. **for**(**int** i=2012;i<year;i++)
73. {
74. **if**(r(i))
75. ans+=1;
76. }
77. **if**(year!=2011)
78. ans+=(year-2012)\*365;
79. ans+=50;
80. **if**(r(year))
81. {
82. **for**(**int** i=0;i<month-1;i++)
83. ans+=num2[i];
84. ans+=day;
85. }
86. **else**
87. {
88. **for**(**int** i=0;i<month-1;i++)
89. ans+=num1[i];
90. ans+=day;
91. }
92. printf("%d\n",change[ans%7]);
93. }
94. }
95. }

40.  [**概率计算**](http://blog.csdn.net/buctyyzyn/article/details/43998125)

问题描述

　　生成n个∈[a,b]的随机整数，输出它们的和为x的概率。

输入格式

　　一行输入四个整数依次为n，a，b，x，用空格分隔。

输出格式

　　输出一行包含一个小数位和为x的概率，小数点后保留四位小数

样例输入

2 1 3 4

样例输出

0.3333

数据规模和约定

　　对于50%的数据，n≤5.

　　对于100%的数据，n≤100,b≤100.

# 解题思路：

这道题是看了别人的思路，自己写的代码，第一次做概率dp的题，没有想法，慢慢学习。

dp[i][j]表示选择i个数，结果为j的概率；

因此递推方程为i，j的前一种状态的和

dp[i][j]+=dp[i-1][j-k]\*p;(a<=k<=b),p为选出一个数的概率

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43998125)[在CODE上查看代码片](https://code.csdn.net/snippets/609606)

1. </pre><pre name="code" **class**="cpp">#include<cstdio>
2. #include<cstring>
3. #include<algorithm>
4. **using** **namespace** std;
5. **double** dp[101][10001]={0};
6. **int** main()
7. {
8. **int** n,a,b,x;
9. scanf("%d%d%d%d",&n,&a,&b,&x);
10. **double** p=1.0/(b-a+1);
11. **for**(**int** i=1;i<=n;i++)
12. {
13. **int** t=min(i\*b,x);
14. **for**(**int** j=i\*a;j<=t;j++)
15. {
16. **if**(i==1)
17. dp[i][j]=p;
18. **else**
19. {
20. **for**(**int** k=a;k<=b;k++)
21. {
22. **if**(k<=j)
23. dp[i][j]+=dp[i-1][j-k]\*p;
24. }
25. }
26. }
27. }
28. printf("%.4lf\n",dp[n][x]);
29. }

41.[6-17复数四则运算](http://blog.csdn.net/buctyyzyn/article/details/43998651)

设计复数库，实现基本的复数加减乘除运算。  
　　输入时只需分别键入实部和虚部，以空格分割，两个复数之间用运算符分隔；输出时按a+bi的格式在屏幕上打印结果。参加样例输入和样例输出。  
　　注意考虑特殊情况，无法计算时输出字符串"error"。

样例输入

2 4 \* -3 2

样例输出

-14-8i

样例输入

3 -2 + -1 3

样例输出

2+1i

# 解题思路：

这个也没啥思路，需要注意的是，处理除法的时候用%g这个格式，该输出格式，会自动保留小数位的有效位；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/43998651)[在CODE上查看代码片](https://code.csdn.net/snippets/609608)

1. #include<cstdio>
2. **int** main()
3. {
4. **int** a,b,c,d;
5. **char** ch[2];
6. scanf("%d%d%s%d%d",&a,&b,ch,&c,&d);
7. **if**(ch[0]=='+')
8. {
9. **if**(b+d>=0)
10. printf("%d+%di\n",a+c,b+d);
11. **else**
12. printf("%d%di\n",a+c,b+d);
13. }
14. **else** **if**(ch[0]=='-')
15. {
16. **if**(b-d>=0)
17. printf("%d+%di\n",a-c,b-d);
18. **else**
19. printf("%d%di\n",a-c,b-d);
20. }
21. **else** **if**(ch[0]=='\*')
22. {
23. **if**(b\*c+a\*d>=0)
24. printf("%d+%di\n",a\*c-b\*d,b\*c+a\*d);
25. **else**
26. printf("%d%di\n",a\*c-b\*d,b\*c+a\*d);
27. }
28. **else**
29. {
30. **if**(c\*c+d\*d!=0)
31. {
32. **if**(b\*c-a\*d>=0)
33. printf("%g+%gi\n",(a\*c+b\*d)\*1.0/(c\*c+d\*d),(b\*c-a\*d)\*1.0/(c\*c+d\*d));
34. **else**
35. printf("%g%gi\n",(a\*c+b\*d)\*1.0/(c\*c+d\*d),(b\*c-a\*d)\*1.0/(c\*c+d\*d));
36. }
37. **else**
38. printf("error\n");
39. }
40. }

42. [最大乘积](http://blog.csdn.net/buctyyzyn/article/details/44003049)

问题描述

　　对于n个数，从中取出m个数，如何取使得这m个数的乘积最大呢？

输入格式

　　第一行一个数表示数据组数  
　　每组输入数据共2行：  
　　第1行给出总共的数字的个数n和要取的数的个数m，1<=n<=m<=15，  
　　第2行依次给出这n个数，其中每个数字的范围满足:a[i]的绝对值小于等于4。

输出格式

　　每组数据输出1行，为最大的乘积。

样例输入

1  
5 5  
1 2 3 4 2

样例输出

48

# 解题思路：

开始想用背包做，但是负数的情况处理不了，后来就用dfs深搜每一种情况过了；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/44003049)[在CODE上查看代码片](https://code.csdn.net/snippets/609792)

1. #include<cstdio>
2. #include<cstring>
3. #include<algorithm>
4. **using** **namespace** std;
5. **int** num[16];
6. **bool** vis[16];
7. **int** n,m;
8. **int** ans;
9. **void** dfs(**int** tt,**int** count)
10. {
11. **if**(count==m)
12. {
13. ans=max(tt,ans);
14. **return** ;
15. }
16. **for**(**int** i=1;i<=n;i++)
17. {
18. **if**(!vis[i])
19. {
20. vis[i]=**true**;
21. dfs(tt\*num[i],count+1);
22. vis[i]=**false**;
23. }
24. }
25. }
26. **int** main()
27. {
29. **int** t;
30. scanf("%d",&t);
31. **while**(t--)
32. {
33. scanf("%d%d",&n,&m);
34. **for**(**int** i=1;i<=n;i++)
35. scanf("%d",&num[i]);
36. memset(vis,0,**sizeof**(vis));
37. ans=-100000;
38. dfs(1,0);
39. printf("%d\n",ans);
40. }
41. }

# [最小方差生成树](http://blog.csdn.net/buctyyzyn/article/details/44020057)

问题描述

给定带权无向图，求出一颗方差最小的生成树。

输入格式

输入多组测试数据。第一行为N,M，依次是点数和边数。接下来M行，每行三个整数U,V,W，代表连接U,V的边，和权值W。保证图连通。n=m=0标志着测试文件的结束。

输出格式

对于每组数据，输出最小方差，四舍五入到0.01。输出格式按照样例。

样例输入

4 5  
1 2 1  
2 3 2  
3 4 2  
4 1 1  
2 4 3  
4 6  
1 2 1  
2 3 2  
3 4 3  
4 1 1  
2 4 3  
1 3 3  
0 0

样例输出

Case 1: 0.22  
Case 2: 0.00

数据规模与约定

1<=U,V<=N<=50,N-1<=M<=1000,0<=W<=50。数据不超过5组。

# 解题思路：

首先得声明一点，如果你做不对这道题目也正常，因为蓝桥杯OJ上的的测试数据有问题；

通过枚举可能的平均值，从而将（原来的权值-平均值）作为路径的权值求最小生成树即可；

因为无法保证算出的最小生成树的平均值就是枚举的平均值，所以要判断是否相同，如果相同才能作为答案更新；

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/44020057)

1. #include<cstdio>
2. #include<cmath>
3. #include<algorithm>
4. **using** **namespace** std;
5. **struct** Edge{
6. **int** x;
7. **int** y;
8. **double** cost;
9. **double** w;
10. }edge[10001];
11. **int** set[10001];
12. **double** ans;
13. **inline** **int** find(**int** x)
14. {
15. **int** i,j,r;
16. r=x;
17. **while**(set[r]!=r)
18. {
19. r=set[r];
20. }
21. i=x;
22. **while**(i!=r)
23. {
24. j=set[i];
25. set[i]=r;
26. i=j;
27. }
28. **return** r;
29. }
30. **inline** **void** merge(**int** x,**int** y)
31. {
32. **int** a=find(x);
33. **int** b=find(y);
34. **if**(a<b)
35. set[b]=a;
36. **else**
37. set[a]=b;
38. }
39. **bool** cmp(**const** Edge &a,**const** Edge &b)
40. {
41. **return** a.w<b.w;
42. }
43. **void** kruskal(**int** n,**int** m,**int** tt)
44. {
45. **int** num,from,to;
46. **double** sum;
47. **double** sum1;
48. **double** t=tt\*1.0/(n-1);
49. **for**(**int** i=0;i<m;i++)
50. edge[i].w=(edge[i].cost-t)\*(edge[i].cost-t);
51. sort(edge,edge+m,cmp);
52. sum1=sum=num=0;
53. **for**(**int** i=0;i<m;i++)
54. {
55. from=find(edge[i].x);
56. to=find(edge[i].y);
57. **if**(from!=to)
58. {
59. merge(from,to);
60. sum+=edge[i].cost;
61. sum1+=edge[i].w;
62. num++;
63. }
64. **if**(num==n-1)
65. **break**;
66. }
67. **if**((**int**)sum==(**int**)tt)
68. ans=min(ans,sum1);
69. }
70. **void** init(**int** n)
71. {
72. **for**(**int** i=1;i<=n;i++)
73. set[i]=i;
74. }
75. **int** main()
76. {
77. **int** n,m;
78. **int** count=1;
79. **int** num[10010];
80. **while**(scanf("%d%d",&n,&m),n||m)
81. {
82. **int** minn=0,maxn=0;
83. **for**(**int** i=0;i<m;i++)
84. {
85. scanf("%d%d%lf",&edge[i].x,&edge[i].y,&edge[i].cost);
86. num[i]=edge[i].cost;
87. }
88. sort(num,num+m);
89. **for**(**int** i=0;i<n-1;i++) minn+=num[i];
90. **for**(**int** i=m-1;i>m-n;i--) maxn+=num[i];
91. ans=0x3f3f3f3f;
92. **for**(**int** i=minn;i<=maxn;i++)
93. {
94. init(n);
95. kruskal(n,m,i);
96. }
97. ans=ans/(n-1);
98. printf("Case %d: %.2lf\n",count++,ans);
99. }
100. }

43.  [**道路和航路**](http://blog.csdn.net/buctyyzyn/article/details/44040497)

1. #include<cstdio>
2. #include<cstring>
3. #include<queue>
4. #include<algorithm>
5. **using** **namespace** std;
6. #define maxn 100010
7. #define inf 1<<30
8. **struct** Edge{
9. **int** w;
10. **int** to;
11. **int** next;
12. }edge[3\*maxn];
13. **int** head[maxn];
14. **long** **long** dis[maxn];
15. **int** vis[maxn];
16. **int** cnt;
17. **int** n,r,p,s;
18. **void** addedge(**int** u,**int** v,**int** w)
19. {
20. edge[cnt].to=v;
21. edge[cnt].w=w;
22. edge[cnt].next=head[u];
23. head[u]=cnt++;
24. }
25. **void** init()
26. {
27. cnt=0;
28. memset(vis,0,**sizeof**(vis));
29. memset(head,-1,**sizeof**(head));
30. memset(edge,0,**sizeof**(edge));
31. }
32. **void** spfa()
33. {
34. **for**(**int** i=1;i<=n;i++) dis[i]=inf;
35. dis[s]=0;
36. vis[s]=1;
37. queue<**int** >q;
38. **while**(!q.empty()) q.pop();
39. q.push(s);
40. **while**(!q.empty())
41. {
42. **int** t=q.front();
43. q.pop();
44. vis[t]=**false**;
45. **for**(**int** i=head[t];i!=-1;i=edge[i].next)
46. {
47. **int** v=edge[i].to;
48. **if**(dis[t]!=inf&&dis[t]+edge[i].w<dis[v])
49. {
50. dis[v]=dis[t]+edge[i].w;
51. **if**(!vis[v])
52. {
53. q.push(v);
54. vis[v]=**true**;
55. }
56. }
57. }
58. }
59. }
60. **int** main()
61. {
62. scanf("%d%d%d%d",&n,&r,&p,&s);
63. init();
64. **int** a,b,c;
65. **for**(**int** i=0;i<r;i++)
66. {
67. scanf("%d%d%d",&a,&b,&c);
68. addedge(a,b,c);
69. addedge(b,a,c);
70. }
71. **for**(**int** i=0;i<p;i++)
72. {
73. scanf("%d%d%d",&a,&b,&c);
74. addedge(a,b,c);
75. }
76. spfa();
77. **for**(**int** i=1;i<=n;i++)
78. {
79. **if**(dis[i]!=inf)
80. printf("%lld\n",dis[i]);
81. **else**
82. printf("NO PATH\n");
83. }
84. }

44.  [**采集金属**](http://blog.csdn.net/buctyyzyn/article/details/44051645)

问题描述

人类在火星上发现了一种新的金属！这些金属分布在一些奇怪的地方，不妨叫它节点好了。一些节点之间有道路相连，所有的节点和道路形成了一棵树。一共有 n 个节点，这些节点被编号为 1~n 。人类将 k 个机器人送上了火星，目的是采集这些金属。这些机器人都被送到了一个指定的着落点， S 号节点。每个机器人在着落之后，必须沿着道路行走。当机器人到达一个节点时，它会采集这个节点蕴藏的所有金属矿。当机器人完成自己的任务之后，可以从任意一个节点返回地球。当然，回到地球的机器人就无法再到火星去了。我们已经提前测量出了每条道路的信息，包括它的两个端点 x 和 y，以及通过这条道路需要花费的能量 w 。我们想花费尽量少的能量采集所有节点的金属，这个任务就交给你了。

输入格式

第一行包含三个整数 n, S 和 k ，分别代表节点个数、着落点编号，和机器人个数。

接下来一共 n-1 行，每行描述一条道路。一行含有三个整数 x, y 和 w ，代表在 x 号节点和 y 号节点之间有一条道路，通过需要花费 w 个单位的能量。所有道路都可以双向通行。

输出格式

输出一个整数，代表采集所有节点的金属所需要的最少能量。

样例输入

6 1 3  
1 2 1  
2 3 1  
2 4 1000  
2 5 1000  
1 6 1000

样例输出

3004

样例说明

所有机器人在 1 号节点着陆。

第一个机器人的行走路径为 1->6 ，在 6 号节点返回地球，花费能量为1000。

第二个机器人的行走路径为 1->2->3->2->4 ，在 4 号节点返回地球，花费能量为1003。

第一个机器人的行走路径为 1->2->5 ，在 5 号节点返回地球，花费能量为1001。

数据规模与约定

本题有10个测试点。

对于测试点 1~2 ， n <= 10 ， k <= 5 。

对于测试点 3 ， n <= 100000 ， k = 1 。

对于测试点 4 ， n <= 1000 ， k = 2 。

对于测试点 5~6 ， n <= 1000 ， k <= 10 。

对于测试点 7~10 ， n <= 100000 ， k <= 10 。

道路的能量 w 均为不超过 1000 的正整数。

# 解题思路：

dp[i][j]表示以i为根的子树停留j个机器人花费的最小值；

然后枚举，i的子孙结点停留0-j个机器人的花费取出最小值即可；（注意这里是子节点，因为是从叶子递推到根，从而结果存储的实际是子树）

**[cpp]** [view plaincopy](http://blog.csdn.net/buctyyzyn/article/details/44051645)

1. #include<cstdio>
2. #include<algorithm>
3. #include<cstring>
4. **using** **namespace** std;
5. #define maxn 100010
6. **int** head[maxn];
7. **int** cnt;
8. **bool** vis[maxn];
9. **int** dp[maxn][11];
10. **int** n,s,sum;
11. **struct** Edge
12. {
13. **int** to;
14. **int** w;
15. **int** next;
16. }edge[3\*maxn];
17. **void** init()
18. {
19. cnt=0;
20. memset(vis,0,**sizeof**(vis));
21. memset(head,-1,**sizeof**(head));
22. }
23. **void** addedge(**int** u,**int** v,**int** c)
24. {
25. edge[cnt].to=v;
26. edge[cnt].w=c;
27. edge[cnt].next=head[u];
28. head[u]=cnt++;
29. }
30. **void** treedp(**int** s)
31. {
32. **for**(**int** i=sum;i>=0;i--)
33. dp[s][i]=0;
34. vis[s]=1;
35. **for**(**int** i=head[s];i!=-1;i=edge[i].next)
36. {
37. **int** v=edge[i].to;
38. **if**(vis[v])
39. **continue**;
40. treedp(v);
41. **for**(**int** j=sum;j>=0;j--)
42. {
43. dp[s][j]+=dp[v][0]+2\*edge[i].w;//儿子结点不放机器人，因为要访问所有结点，所以至少有一个机器人去了儿子结点又返回父亲结点
44. **for**(**int** k=1;k<=j;k++)
45. dp[s][j]=min(dp[s][j],dp[s][j-k]+k\*edge[i].w+dp[v][k]);//枚举儿子结点放几个机器人
46. }
47. }
48. }
49. **int** main()
50. {
51. init();
52. scanf("%d%d%d",&n,&s,&sum);
53. **for**(**int** i=0;i<n-1;i++)
54. {
55. **int** a,b,c;
56. scanf("%d%d%d",&a,&b,&c);
57. addedge(a,b,c);
58. addedge(b,a,c);
59. }
60. treedp(s);
61. printf("%d\n",dp[s][sum]);
62. }

原作者博客：<http://blog.csdn.net/buctyyzyn>。

收集人：忘川半夜。