

双向搜索

简介&思想

在进行搜索的时候可以从某一个角度将搜索的状态减少或减小得到同样的状态的时间开销，从而减少了时间复杂度。

例题

世界冰球锦标赛

来源：洛谷 p4799

概述：共 N ($1 \leq N \leq 40$) 场比赛，每场比赛的门票有一定的价格，给定 M 元，计算可能的观看方案。

答案：分别计算前 $N/2$ 场和后 $N/2$ 场的方案，分别只有 $2^{20}=1,048,576$ 种情况，然后分别排序，在两个集合中各用一个指针，利用单调性在线性时间内得到答案。

```
#include<cstdio>
#include<algorithm>
#include<queue>
#define ll long long
using namespace std;
int n,half,tot1,tot2;
ll m,a[50],ans,ll[2100000],l2[2100000],times[2100000];
void dfs1(int x,ll cost)
{
    if(x>half)
    {
        ll[++tot1]=cost;
        return;
    }
    if(cost+a[x]<=m)
        dfs1(x+1,cost+a[x]);
    dfs1(x+1,cost);
}
void dfs2(int x,ll cost)
{
    if(x>n)
    {
        l2[++tot2]=cost;
        return;
    }
    if(cost+a[x]<=m)
        dfs2(x+1,cost+a[x]);
    dfs2(x+1,cost);
}
```

```
}
int main()
{
    scanf("%d%lld",&n,&m);
    for(int i=1;i<=n;i++)
        scanf("%lld",&a[i]);
    half=n/2;
    dfs1(1,0);
    dfs2(half+1,0);
    sort(l1+1,l1+tot1+1);
    sort(l2+1,l2+tot2+1);
    int x=0;
    l2[0]=-1;
    ll sum;
    ans=sum=tot2;
    for(int i=1;i<=tot2;i++)
    {
        if(l2[i]+l1[1]>m)
        {
            ans=sum=i-1;
            break;
        }
        if(l2[i]>l2[i-1])
            l2[++x]=l2[i];
        times[x]++;
    }
    int last;
    last=tot2=x;
    for(int i=2;i<=tot1;i++)
    {
        if(l1[i]>l1[i-1])
        {
            int j=last;
            for(;j>0;j--)
            {
                if(l2[j]+l1[i]<=m)
                    break;
                sum-=times[j];
            }
            if(!j)
                break;
            last=j;
        }
        ans+=sum;
    }
    printf("%lld",ans);
    return 0;
}
```

Knights of Ni S

来源：洛谷 p5195

概述：贝茜需要先到达灌木丛，再到达骑士所在地，前一个过程中不能通过骑士所在地。给出一个地图，标记了可以走、不可以走的位置，以及贝茜的初始位置、灌木丛位置、骑士所在位置。计算贝茜所走的最短距离。

答案：用bfs分别得出贝茜和骑士到达每一个灌木丛的最短距离，加和最小的即为答案，注意前一个距离判断时不能通过骑士所在地。

```
#include<cstdio>
#include<queue>
#include<cstring>
#define inf 0x3f3f3f3f
using namespace std;
int
w,h,grid[1005][1005],dis1[1005][1005],dis2[1005][1005],vist[1005][1005],dx[
4]={0,0,-1,1},dy[4]={1,-1,0,0},d1,d2,ans=inf,x1,y1,x2,y2;
struct unit{int x,y;};
void bfs()
{
    memset(dis1,0x3f,sizeof(dis1));
    memset(dis2,0x3f,sizeof(dis2));
    queue<unit>q;
    dis1[x1][y1]=0;
    vist[x1][y1]=1;
    q.push({x1,y1});
    int m=1;
    unit head;
    while(m-->0)
    {
        head=q.front();
        q.pop();
        for(int i=0,x_,y_;i<4;i++)
        {
            x_=head.x+dx[i],y_=head.y+dy[i];
            if(x_&&x_<=h&&y_&&y_<=w&&grid[x_][y_]!=1&&!vist[x_][y_]&&grid[x_][y_]!=3)
            {
                dis1[x_][y_]=dis1[head.x][head.y]+1;
                vist[x_][y_]=1;
                m++;
                q.push({x_,y_});
            }
        }
    }
    memset(vist,0,sizeof(vist));
    dis2[x2][y2]=0;
```

```
vist[x2][y2]=1;
q.push({x2,y2});
m=1;
while(m--)
{
    head=q.front();
    q.pop();
    if(grid[head.x][head.y]==4)
        ans=min(ans,dis1[head.x][head.y]+dis2[head.x][head.y]);
    for(int i=0,x_,y_;i<4;i++)
    {
        x_=head.x+dx[i],y_=head.y+dy[i];
        if(x_&&x_<=h&&y_&&y_<=w&&grid[x_][y_]!=1&&!vist[x_][y_])
        {
            vist[x_][y_]=1;
            dis2[x_][y_]=dis2[head.x][head.y]+1;
            m++;
            q.push({x_,y_});
        }
    }
}
}
int main()
{
    scanf("%d%d",&w,&h);
    int m=0;
    for(int i=1;i<=h;i++)
        for(int j=1;j<=w;j++)
        {
            scanf("%d",&grid[i][j]);
            if(grid[i][j]==2)
                x1=i,y1=j;
            else if(grid[i][j]==3)
                x2=i,y2=j;
        }
    bfs();
    printf("%d",ans);
    return 0;
}
```

Balanced Cow Subsets G

来源：洛谷 p3067

概述：给n个数，从中任意选出一些数，使这些数能分成和相等的两组。求有多少种选数的方案。

答案：每一个数分成三种状态：不加入集合、加入第一个集合、加入第二个集合。在一个子集中，加入第一个集合记为加这个数，加入第二个集合记为减这个数，那么一个集合的sum为0即符合条件（空集除外）。分别得出前半、后半的选择情况，排序后利用指针简化组合过程。

```
#include<cstdio>
#include<algorithm>
using namespace std;
int n,m[25],half,tot1,tot2,po[25],ans,vist[11000000];
struct unit
{
    int state,val;
    bool operator < (const unit&x) const
    {
        if(val!=x.val) return val<x.val;
        else return state<x.state;
    }
}set1[60000],set2[60000];
void dfs1(int x,int state,int sum)
{
    if(x>half)
    {
        set1[++tot1]={state,sum};
        return;
    }
    dfs1(x+1,state,sum);
    dfs1(x+1,state|po[x],sum+m[x]);
    dfs1(x+1,state|po[x],sum-m[x]);
}
void dfs2(int x,int state,int sum)
{
    if(x>n)
    {
        set2[++tot2]={state,sum};
        return;
    }
    dfs2(x+1,state,sum);
    dfs2(x+1,state|po[x],sum+m[x]);
    dfs2(x+1,state|po[x],sum-m[x]);
}
int main()
{
    for(int i=1,j=1;i<=20;i++,j<=1)
        po[i]=j;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        scanf("%d",&m[i]);
    half=n/2;
    dfs1(1,0,0);
    dfs2(half+1,0,0);
    sort(set1+1,set1+tot1+1);
    sort(set2+1,set2+tot2+1);
    /*for(int i=1;i<=tot1;i++)
        printf("%d ",set1[i].val);
    printf("\n");*/
}
```

```
for(int i=1;i<=tot2;i++)
    printf("%d ",set2[i].val);
printf("\n%d %d\n",tot1,tot2);*/
int l=1,r=tot2,r0=tot2;
while(l<=tot1&&r0)
{
    //printf("%d %d %d %d\n",l,r,r0,ans);
    if(set1[l].val==set1[l-1].val)
    {
        if(set1[l].state==set1[l-1].state)
        {
            l++;
            continue;
        }
        for(int i=r+1;i<=r0;i++)
            if(!vist[set1[l].state|set2[i].state])
            {
                vist[set1[l].state|set2[i].state]=1;
                ans++;
            }
    }
    else
    {
        for(r0=r;r0>0;r0--)
            if(set2[r0].val+set1[l].val<=0)
                break;
        r=r0;
        if(!(set2[r0].val+set1[l].val))
        {
            for(;r>0;r--)
                if(set2[r].val+set1[l].val<0)
                    break;
            for(int i=r+1;i<=r0;i++)
                if(!vist[set1[l].state|set2[i].state])
                {
                    vist[set1[l].state|set2[i].state]=1;
                    ans++;
                }
        }
    }
    l++;
}
printf("%d",ans-1);
return 0;
}
```

总结

搜索中常用的技巧，在合适的场合使用能有效降低时间开销。

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:no_morning_training:shaco:%E7%9F%A5%E8%AF%86%E7%82%B9:%E6%90%9C%E7%B4%A2:%E5%8F%8C%E5%90%91%E6%90%9C%E7%B4%A2

Last update: 2020/08/14 15:57