dfs

bfs

剪枝

折半搜索meet in middle

1.n个数，选取一些数和小于<=m的总方案数，$n \le 40$

$O(2^{n/2})$，相当于开根号

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int N=22;
int n,m;
int a[N<<1];
int b[1<<N],c[1<<N];
void dfs(int now,int n,int val,int s[],int &cnt){
    if (now>n){
        s[++cnt]=val;
        return;
    }
    dfs(now+1,n,val,s,cnt);
    dfs(now+1,n,val+a[now],s,cnt);
}
signed main(){
    cin>>n>>m;
    for (int i=1;i<=n;i++)cin>>a[i];
    int cntb=0,cntc=0;
    dfs(1,n/2,0,b,cntb);
    dfs(n/2+1,n,0,c,cntc);
    int ans=0;
    sort(b+1,b+1+cntb);
    for (int i=1;i<=cntc;i++){
        ans+=upper_bound(b+1,b+cntb+1,m-c[i])-b-1;
    }
    cout<<ans<<endl;
    return 0;
}
```

2.n个数选出一些数，分成两份，两份和相等，有多少选数方案 $n \le 20$

```cpp
#include<bits/stdc++.h>
```

```
using namespace std;
#define int long long
const int N=2e3+5,mod=1e9+7;
int n,m,k;
int a[N];
unordered_map<int,bitset<N>>p;
bitset<N>vis[N];
int ans=0,mid;
void dfs1(int now,int n,int sum,int state){
    if (now>n){
        p[sum].set(state);
        return;
    }
    dfs1(now+1,n,sum,state);//不放
    dfs1(now+1,n,sum+a[now],state|(1<<(now-1)));//放左边
    dfs1(now+1,n,sum-a[now],state|(1<<(now-1)));//放右边
}
void dfs2(int now,int n,int sum,int state){
    if (now>n){
        if (p.count(-sum)){
            bitset<N>s(p[-sum]);
            s&=~vis[state];
            ans+=s.count();
            vis[state]|=s;//记录重复的
        }
        return;
    }
    dfs2(now+1,n,sum,state);
    dfs2(now+1,n,sum+a[now],state|(1<<(now-1-mid)));
    dfs2(now+1,n,sum-a[now],state|(1<<(now-1-mid)));
}
signed main(){
    cin>>n;
    for (int i=1;i<=n;i++)scanf("%d",&a[i]);
    mid=n/2;
    dfs1(1,mid,0,0);dfs2(mid+1,n,0,0);
    cout<<ans-1<<endl;//0 0不算
    return 0;
}
```

A*

f=h+g

洛谷P1379八数码

```
#include<bits/stdc++.h>
using namespace std;
string goal="123804765",start;
int h(string s){//h函数
    int sum=0;
    for (int i=0;i<9;i++){
        sum+=(s[i]!=goal[i]);
    }
    return sum;
}
struct node{
    int f,steps;
    string s;
    node(int v,int st,string ss){
        f=v;steps=st;s=ss;
    }
    bool operator<(const node a)const {
        if (f==a.f)return steps>a.steps;
        return f>a.f;
    }
};//f函数值
map<string,bool>p;
signed main(){
    cin>>start;
    if (h(start)==0) {puts("0"); return 0;}
    priority_queue<node>q;
    q.push({h(start),0,start});
    p[start]=1;
    while (!q.empty()){
        node t=q.top();q.pop();
        string s=t.s;
        if (s==goal){
            cout<<t.steps<<endl;return 0;
        }
        for (int i=0;i<9;i++){
            if (s[i]=='0'){
                int l=i/3,r=i%3;
                if (l>0){
                    swap(s[i],s[i-3]);
                    if (!p[s]){
                        q.push({h(s)+t.steps+1,t.steps+1,s});
                        p[s]=1;
                    }
                    swap(s[i],s[i-3]);
                }
                if (l<2){
                    swap(s[i],s[i+3]);
                    if (!p[s]){
                        q.push({h(s)+t.steps+1,t.steps+1,s});
                        p[s]=1;
                    }
                    swap(s[i],s[i+3]);
                }
                if (r>0){
                    swap(s[i],s[i-1]);
                    if (!p[s]){
                        q.push({h(s)+t.steps+1,t.steps+1,s});
                        p[s]=1;
                    }
                    swap(s[i],s[i-1]);
                }
                if (r<2){
                    swap(s[i],s[i+1]);
                    if (!p[s]){
                        q.push({h(s)+t.steps+1,t.steps+1,s});
                        p[s]=1;
                    }
                    swap(s[i],s[i+1]);
                }
                break;
            }
        }
    }
    return 0;
}
```

IDA*

像 BFS 那样，每次只扩展一层节点，却采用 DFS 方式来遍历搜索树，这就是迭代加深搜索。

再加上一个估价函数来减小搜索量，就是 IDA*了。

洛谷P1379八数码

```
#include<bits/stdc++.h>
using namespace std;
string goal="123804765",start;
int dep;
int h(string s){
    int sum=0;
    for (int i=0;i<9;i++){
        sum+=(s[i]!=goal[i]);
    }
    return sum;
}
bool dfs(int depth,string s,int pre) {
    if (h(s)==0)return true;
    if (depth+h(s)-1>dep)return false;
    for (int i=0;i<9;i++){
        if (s[i]=='0'){
            int l=i/3,r=i%3;
            if (l>0&&pre!=2){
                swap(s[i],s[i-3]);
                if (dfs(depth+1,s,1))return true;
                swap(s[i],s[i-3]);
            }
            if (l<2&&pre!=1){
                swap(s[i],s[i+3]);
                if (dfs(depth+1,s,2))return true;
                swap(s[i],s[i+3]);
            }
            if (r>0&&pre!=4){
                swap(s[i],s[i-1]);
                if (dfs(depth+1,s,3))return true;
                swap(s[i],s[i-1]);
            }
            if (r<2&&pre!=3){
                swap(s[i],s[i+1]);
                if (dfs(depth+1,s,4))return true;
                swap(s[i],s[i+1]);
            }
            break;
        }
    }
    return false;
}
signed main(){
    cin>>start;
    if (h(start) == 0) {puts("0"); return 0;}
    for (dep=1;;dep++){
        if (dfs(0,start,-1)){
            printf("%d\n",dep);
            return 0;
        }
    }
    return 0;
}
```

DLX