**2-3 半数集问题**

#include<bits/stdc++.h>

using namespace std;

int cnt = 1;//自身

void slove(int m)

{

if(m <= 1)

return;

for(int i=1; i<=m/2; i++)

{

cnt++;

slove(i);

}

}

int main()

{

int n;

cin>>n;

slove(n);

cout<<cnt<<endl;

return 0;

}

**2-5 有重复元素的排列问题**

#include<bits/stdc++.h>

using namespace std;

const int maxn = 1000+5;

char arr[maxn];

int sum = 0;

void myswap(char &a,char &b)

{

char temp;

temp = a;

a = b;

b = temp;

}

int isOk(char arr[],int k,int i)

{

//看第r个元素是否在前面元素[k,````i-1]中出现过

if(i > k)

{

for(int j=k; j<i; j++)

{

if(arr[j]==arr[i])

return 0;

}

}

return 1;

}

void perm(char arr[],int k, int m)

{

if(k == m)

{

sum++;

for(int i=0; i<=m; i++)

cout<<arr[i];

cout<<endl;

}

for(int i=k; i<=m; i++)

{

if(isOk(arr,k,i))

{

myswap(arr[k],arr[i]);

perm(arr,k+1,m);

myswap(arr[k],arr[i]);

}

}

}

int main()

{

int n;

cin>>n;

for(int i=0; i<n; i++)

cin>>arr[i];

perm(arr,0,n-1);

cout<<sum<<endl;

return 0;

}

**2-6 排列的字典序问题**

long long n;

Cin>>n;

int f(int n)//求阶乘

{

if(n==1) return 1;

return n\*f(n-1);

}

int main()

{

long long a[1000];

long long sum=0;

for(i=0;i<n;i++) cin>>a[i];

for(i=0;i<n-1;i++)

{

long long count=0;

for(j=i+1;j<n;j++)

{

if(a[j]<a[i]) count++;

}

sum+=(count\*f(n-i-1));

}

cout<<sum<<endl;

if(next\_permutation(a,a+n))

{

for(int i=0;i<n;i++) cout<<a[i]<<" ";

cout<<endl;

}

return 0;

}

**2-10 标准二维表问题**

#include<bits/stdc++.h>

using namespace std;

int f(int n)

{

int count = 0;

if(n==1 || n==0)

return 1;

for(int i=0;i<n;i++)

count = count + f(i) \* f(n-1-i);

return count;

}

int main()

{

int n;

cin>>n;

//printf("%d\n",f(n));

cout<<f(n)<<endl;

return 0;

}

4-1 会场安排问题

#include <bits/stdc++.h>

using namespace std;

const int maxn = 1000+5;

struct job

{

int begin,end;

bool ok;

};

int manage(int n,job \*b)

{

int count=n,roomavail=0,roomnum=0;//count表示还剩余几个活动未安排

while(count>0)

{

for(int i=1;i<=n;i++)

{

if((b[i].begin>roomavail)&&(b[i].ok==0))

{

roomavail=b[i].end;//将当前活动加入该会场，并且更新会场的空闲时间

b[i].ok=1;//标记该活动已被安排

count--;//未安排的活动数减1

}

}

roomavail=0;//把roomavail初始化，相当于重新再找一个房间

roomnum++;//遍历一次，使用的会场数加1

}

return roomnum;//返回使用的会场数

}

bool cmp(job x,job y)

{

return x.end < y.end;

}

int main()

{

int n; cin>>n;

job b[n+1];

b[0].begin=0;

b[0].end=0;

b[0].ok=0;

for(int i=1;i<=n;i++)

{

cin>>b[i].begin>>b[i].end;

b[i].ok=0;

}

sort(b+1,b+n+1,cmp);

int ans = manage(n,b);

cout<<ans<<endl;

return 0;

}

**4-5 程序存储问题**

#include<bits/stdc++.h>

using namespace std;

const int maxn = 1000+10;

int a[maxn];

int Greedy(int a[],int n,int L)

{

sort(a,a+n);

int temp = 0;

for(int i=0; i<n; i++){//最后一个元素为下标n-1的元素

temp = temp + a[i];

if (temp>L) return i;

if(i==(n-1)) return n; //没有想到还有存得完的情况。

}

}

int main()

{

int n,l;

cin>>n>>l;

for(int i=0; i<n; i++) cin>>a[i];

cout<<Greedy(a,n,l)<<endl;;

return 0;

}

**4-6 最优服务次序问题**

#include<bits/stdc++.h>

using namespace std;

const int maxn = 1000+10;

double a[maxn];

int Greedy(double a[],int n)

{

double sum = 0;

for(int i=0; i<n; i++)

sum += a[i]\*(n-i);

return sum/n;

}

int main()

{

int n;

cin>>n;

for(int i=0; i<n; i++)

cin>>a[i];

sort(a,a+n);

cout<<Greedy(a,n)<<endl;;

return 0;

}

**4-9 汽车加油问题**

#include<bits/stdc++.h>

using namespace std;

const int maxn = 1000+10;

int a[maxn];

int main()

{

int n,k;

cin>>n>>k;

for(int i=1; i<=k+1; i++)

cin>>a[i];

int count=0;

int sum=0;

for(int i=1;i<=k+1;i++)

{

sum+=a[i];

if(sum>=n)

{

sum=a[i];

count++;

// cout<<"第"<<count <<"次加油的位置："<<a[i-1]<<endl;

}

}

cout << "总共加油："<<count<<"次"<< endl;

return 0;

}

4-10 区间覆盖问题

#include<bits/stdc++.h>

using namespace std;

const int maxn = 1000+10;

int a[maxn];

int main()

{

int n,k;

cin>>n>>k;

for(int i=0; i<n; i++)

cin>>a[i];

sort(a,a+n);

int sum = 1 ;

int temp;

for(int i=0, temp=a[0]; i<n; i++)

{

if(a[i]-temp>k)

{

temp = a[i] ;

sum ++ ;

}

}

cout << sum << endl ;

return 0;

}

4-14 多元Huffman问题

#include<bits/stdc++.h>

using namespace std;

const int maxn = 1000+10;

int stone[maxn];

int stonemax(int n,int k)

{

priority\_queue<int> q;//生成最大优先队列

for(int i=0;i<n;i++)

q.push(stone[i]);

int sum=0,max=0;

while(q.size()>2)//因为要求最大总费用，所以要求合并次数尽量多

{

sum=0;

for(int i=0;i<2;i++)

{

sum+=q.top();

q.pop();

}

max+=sum;

q.push(sum);

}

while(!q.empty())

{

max+=q.top();

q.pop();

}

return max;

}

int stonemin(int n,int k)

{

priority\_queue<int,vector<int>,greater<int> > q;//生成最小优先队列

for(int i=0;i<n;i++)

q.push(stone[i]);

int sum=0,min=0;

while(q.size()>k)//因为要求最小总费用，所以要求合并次数尽量少

{

sum=0;

for(int i=0;i<k;i++)

{

sum+=q.top();

q.pop();

}

min+=sum;

q.push(sum);

}

while(!q.empty())

{

min+=q.top();

q.pop();

}

return min;

}

int main()

{

int n,k;

cin>>n>>k;

for(int i=0; i<n; i++)

cin>>stone[i];

cout<<stonemax(n,k)<<endl;

cout<<stonemin(n,k)<<endl;

return 0;

}

**4-15 最优分解问题**

#include<bits/stdc++.h>

using namespace std;

int f(int n)

{

int a[100];//保存分解后的数

int k=1,sum=1;//k为数组a索引，sum乘积

a[1]=2; //先减2,注意这里索引是1

n-=2;

while(n>a[k])

{

k++;

a[k]=a[k-1]+1;

n-=a[k];

}

if(n==a[k])

{

a[k]++; n--;

}

for(int i=0;i<n;i++)

a[k-i]++;

for(int j=1;j<=k;j++)//从1开始

sum\*=a[j];

return sum;

}

int main()

{

int n; cin>>n;

cout<<f(n)<<endl;

return 0;

}

6-6