1a.bubble sort

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

int main(){

int n=0,s,size=100,temp;

long long int c;

char str[15];

int \*arr = (int\*)calloc(size,sizeof(int));

scanf("%d",&s);

strcpy(str,"input1.txt");

FILE \*fp;

for(int i=1; i<=s; i++){

str[5]=48+i ;

fp = fopen(str,"r");

n=0;

c=0;

while(fscanf(fp,"%d ",&arr[n])>0){

if(n>=size-1){

arr=(int\*)realloc(arr,(size+100)\*sizeof(int));

size=size+100;

}

n++;

}

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

for(int j=0; j<n-i; j++){

if(arr[j]>arr[j+1]){

c++;

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

fclose(fp);

printf("%lld\n",c);

}

return 0;

}

1a)merge sort

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int size = 100;

int \*a,\*b;

long long int c;

void merge(int low,int mid,int high){

int i,j,k=low;

for(i=low,j=mid+1; i<=mid && j<=high; ){

if(a[i]>a[j]){

b[k++] = a[i++];

c++;

}

else{

b[k++] = a[j++];

c++;

}

}

while(i<=mid){

b[k++] = a[i++];

c++;

}

while(j<=high){

b[k++]=a[j++];

c++;

}

for(i=low;i<=high;i++){

a[i]=b[i];

}

}

void mergeSort(int low,int high){

if(high>low){

int mid=(low+high)/2;

mergeSort(low,mid);

mergeSort(mid+1,high);

merge(low,mid,high);

}

}

int main(){

int n=0,s,temp;

char str[15];

a=(int\*)calloc(size,sizeof(int));

b=(int\*)calloc(size,sizeof(int));

scanf("%d",&s);

strcpy(str,"input1.txt");

FILE \*fp;

for(int i=1;i<=s;i++){

str[5]=48+i;

fp=fopen(str,"r");

n=0;

c=0;

while(fscanf(fp,"%d ",&a[n])>0){

if(n>=size-1){

a=(int\*)realloc(a,(size+100)\*sizeof(int));

b=(int\*)realloc(b,(size+100)\*sizeof(int));

size+=100;

}

n++;

}

mergeSort(0,n-1);

fclose(fp);

printf("%lld\n",c);

}

return 0;

}

1a)quick sort

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

long long int c=0;

int size = 100;

int \*arr;

int temp;

int partition(int low, int high){

int pi\_i = low;

int pivot = arr[pi\_i];

int i=low-1, j=high+1;

while(i<j){

do{

i++;

}while(arr[i]<=pivot);

do{

j--;

}while(arr[j]>pivot);

if(i<j){

c++;

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

c++;

temp = arr[pi\_i];

arr[pi\_i] = arr[j];

arr[j] = temp;

return j;

}

void quickSort(int low, int high){

if(low<high){

int pi = partition(low,high);

quickSort(low, pi-1);

quickSort(pi+1, high);

}

}

int main()

{

int n=0,s,temp;

char str[15];

arr=(int\*)calloc(size,sizeof(int));

scanf("%d",&s);

strcpy(str,"input5.txt");

FILE \*fp;

for(int i=1; i<=s; i++){

str[5] = 48+i ;

fp = fopen(str,"r");

n=0;

c=0;

while(fscanf(fp,"%d ",&arr[n])>0){

if(n>=size-1){

arr=(int\*)realloc(arr,(size+100)\*sizeof(int));

size+=100;

}

n++;

}

fclose(fp);

quickSort(0,n-1);

printf("%lld\n",c);

}

return 0;

}

2a) linear search

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void main(){

int n=0,s,size=100,key,num;

char str[15];

FILE \*fp;

strcpy(str,"input1.txt");

scanf("%d",&s);

for(int i=1; i<=s; i++){

str[5] = 48+i;

n=0;

scanf("%d",&key);

fp = fopen(str,"r");

while((fscanf(fp,"%d ",&num))>0){

if(num == key){

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

key = -1;

break;

}

n++;

}

if(key != -1)

printf("%d \n",key);

}

}

2b)binary search

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int binary(int a[],int n,int x){

int mid;

int l=0,u=n-1;

int count=1;

while(l<u){

mid=(l+u)/2;

if(a[mid]>x)

u=mid-1;

if(a[mid]<x)

l=mid+1;

if(a[mid]==x){

break;

}

count++;

}

return count;

}

void sort(int a[],int n){

int i,j;

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

for(j=0;j<n;j++){

if(a[i]<a[j]){

int temp=a[j];

a[j]=a[i];

a[i]=temp;

}

}

}

}

int main(){

FILE\*fp;

fp=fopen("input1.txt","r");

int\*a;

a=(int\*)malloc(5000\*sizeof(int));

int count=0;

while(fscanf(fp,"%d",&a[count])!=EOF)

count++;

int x;

scanf("%d",&x);

scanf("%d",&x);

sort(a,count);

printf("%d",binary(a,count,x));

free(a);

fp=fopen("input2.txt","r");

a=(int\*)malloc(5000\*sizeof(int));

count=0;

while(fscanf(fp,"%d",&a[count])!=EOF)

count++;

scanf("%d",&x);

sort(a,count);

printf("\n%d",binary(a,count,x)+1);

free(a);

fp=fopen("input3.txt","r");

a=(int\*)malloc(5000\*sizeof(int));

count=0;

while(fscanf(fp,"%d",&a[count])!=EOF)

count++;

scanf("%d",&x);

sort(a,count);

printf("\n%d",binary(a,count,x)+1);

free(a);

fp=fopen("input4.txt","r");

a=(int\*)malloc(5000\*sizeof(int));

count=0;

while(fscanf(fp,"%d",&a[count])!=EOF)

count++;

scanf("%d",&x);

sort(a,count);

printf("\n%d",binary(a,count,x)+1);

free(a);

fp=fopen("input5.txt","r");

a=(int\*)malloc(5000\*sizeof(int));

count=0;

while(fscanf(fp,"%d",&a[count])!=EOF)

count++;

scanf("%d",&x);

sort(a,count);

printf("\n%d",binary(a,count,x)-1);

free(a);

return 0;

}

3a) sequential representation

#include<stdio.h>

#include<stdlib.h>

int main()

{

int n,m,k,l,d;

int a[100][100];

scanf("%d%d",&n,&m);

scanf("%d",&d);

if(d==1)

{

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

{

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

{

a[i][j]=0;

}

}

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

{

scanf("%d %d",&k,&l);

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

}

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

{

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

{

printf(" %d ",a[i][j]);

}

printf("\n");

}

}

else

{

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

{

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

{

a[i][j]=0;

}

}

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

{

scanf("%d%d",&k,&l);

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

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

}

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

{

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

{

printf(" %d ",a[i][j]);

}

printf("\n");

}

}

}

3b) find the degree of all the vertices

#include<stdio.h>

void main()

{

int i,j,n,a[20][30],count=0;

scanf("%d",&n);

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

{

for(j=0;j<n;j++)

{

scanf("%d",&a[i][j]);

}

}

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

{

for(j=0;j<n;j++)

{

if(a[i][j]==1)

count++;

}

printf(" %d",count);

count=0;

}

}

4a) multi 2 given square matrices using starenns matrix

#include<stdio.h>

int main()

{

int m[2][2],n[2][2],o[2][2]={0};

int p,q,r,s,t,u,v,i,j;

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

{

for(j=0;j<2;j++)

{

scanf("%d",&m[i][j]);

}

}

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

{

for(j=0;j<2;j++)

{

scanf("%d",&n[i][j]);

}

}

p=(m[0][0]+m[1][1])\*(n[0][0]+n[1][1]);

q=(m[1][0]+m[1][1])\*n[0][0];

r=m[0][0]\*(n[0][1]-n[1][1]);

s=m[1][1]\*(n[1][0]-n[0][0]);

t=(m[0][0]+m[0][1])\*n[1][1];

u=(m[1][0]-m[0][0])\*(n[0][0]+n[0][1]);

v=(m[0][1]-m[1][1])\*(n[1][0]+n[1][1]);

o[0][0]=p+s-t+v;

o[0][1]=r+t;

o[1][0]=q+s;

o[1][1]=p+r-q+u;

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

{

printf("\n");

for(j=0;j<2;j++)

{

printf("%d ",o[i][j]);

}

}

}

5) job sequencing with dead lines

#include<stdio.h>

#include<stdlib.h>

void main(){

int n,k=-1,temp;

scanf("%d",&n);

int P[n],D[n],J[n];

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

J[i] = i+1;

scanf("%d%d",&P[i],&D[i]);

if(k<D[i]){

k = D[i];

}

}

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

int S[k];

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

S[i] = -1;

}

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

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

if(P[j]<P[j+1]){

temp = P[j];

P[j] = P[j+1];

P[j+1] = temp;

temp = D[j];

D[j] = D[j+1];

D[j+1] = temp;

temp = J[j];

J[j] = J[j+1];

J[j+1] = temp;

}

}

}

printf("INDEX\tPROFIT\tDEADLINE\tSLOT ALLOTTED\n");

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

temp = -1;

for(int j=k; j>0; j--){

if(S[(j-1)]==-1 && (D[i]==j||D[i]>j)){

S[(j-1)] = 1;

temp = 1;

printf("J%d\t%d\t%d\t[%d-%d]\n",J[i],P[i],D[i],(j-1),j);

break;

}

}

if(temp == -1){

printf("J%d\t%d\t%d\tREJECTED\n",J[i],P[i],D[i]);

}

}

}

6)knapsack problem using greedy approach

#include<stdlib.h>

void swap(float \*a,float \*b){

float\*temp;

temp=a;

a=b;

b=temp;

}

void main(){

int N;

float K,temp;

//printf("number of items:");

scanf("%d",&N);

scanf("%f",&K);

float P[N],W[N],C[N];

float Pro=0;

for(int i=0;i<N;i++){

//printf("Weight and Profit for item[%d]:",i);

scanf("%f%f",&W[i],&P[i]);

C[i]=P[i]/W[i];

}

//printf("The capacity of Knapsack :");

for(int i=0;i<(N-1);i++){

for(int j=0;j<(N-i-1);j++){

if(C[j+1]>C[j]){

temp=C[j];

C[j]=C[j+1];

C[j+1]=temp;

temp=W[j];

W[j]=W[j+1];

W[j+1]=temp;

temp=P[j];

P[j]=P[j+1];

P[j+1]=temp;

//swap(&C[j],&C[j+1]);

//swap(&P[j],&P[j+1]);

//swap(&W[j],&W[j+1])

}

}

}

//for(int i=0;i<N;i++){

// printf("%f-%f-%f\n",W[i],P[i],C[i]);

//}

for(int i=0;i<N && K>0;i++){

if(K>=W[i]){

Pro=Pro +P[i];

K=K-W[i];

//printf("%f-%f-%f-%f\n",W[i],P[i],K,Pro);

}

else{

Pro=Pro+(K)\*C[i];

K=K-W[i];

}

}

printf("%f",Pro);

}

7a)minimum spanning tree using prims algorithm

#include<stdio.h>

int a,b,u,ve,i,j,n,k=1;

int v[10]={0},c[10][10],s[10][10],min,cost=0;

void main()

{

scanf("%d",&n);

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

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

{

s[i][j]=0;

scanf("%d",&c[i][j]);

if(c[i][j]==0)

c[i][j]=999;

}

v[1]=1;

while(k<n)

{

for(i=1,min=999;i<=n;i++)

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

if(c[i][j]<min)

if(v[i]!=0)

{

min=c[i][j];

a=u=i;

b=ve=j;

}

if(v[u]==0||v[ve]==0)

{

cost+=min;

v[b]=1;

s[a][b]=s[b][a]=min;

k++;

}

c[a][b]=c[b][a]=999;

}

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

{

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

{

printf("%d\t",s[i][j]);

}

printf("\n");

}

printf("Total cost of spanning tree=%d",cost);

}

7b) minimum spanning tree using kruska;s algorithm

#include<stdio.h>

#include<stdlib.h>

int i,j,k,a,b,u,v,n,ne=1;

int min,mincost=0,cost[12][12],p[12],d[12][12];

int find(int);

int uni(int,int);

void main()

{

scanf("%d",&n);

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

{

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

{

d[i][j]=0;

scanf("%d",&cost[i][j]);

if(cost[i][j]==0)

cost[i][j]=999;

}

}

while(ne<n)

{

for(i=1,min=999;i<=n;i++)

{

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

{

if(cost[i][j]<min)

{

min=cost[i][j];

a=u=i;

b=v=j;

}

}

}

u=find(u);

v=find(v);

if(uni(u,v))

{

mincost+=min;

d[a][b]=d[b][a]=min;

ne++;

}

cost[a][b]=cost[b][a]=999;

}

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

{

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

{

printf("%d ",d[i][j]);

}

printf("\n");

}

printf("minimum cost of spanning tree=%d\n",mincost);

}

int find(int i)

{

while(p[i])

i=p[i];

return i;

}

int uni(int i,int j)

{

if(i!=j)

{

p[j]=i;

return 1;

}

return 0;

}

8) all pair shortest path problem

#include<stdio.h>

int min(int a,int b)

{

if(a<b)

return a;

else

return b;

}

int main()

{

int n,i,j,k;

scanf("%d",&n);

int a[n][n];

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

{

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

{

scanf("%d",&a[i][j]);

if(a[i][j]==0&&i!=j)

a[i][j]=999;

}

}

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

{

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

{

for(k=1;k<=n;k++)

{

a[j][k]=min(a[j][k],a[j][i]+a[i][k]);

}

}

}

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

{

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

{

printf("%d ",a[i][j]);

}

printf("\n");

}

}

9)optimal binary search tree using dynamic programming

#include<stdio.h>

void print(int, int);

int min\_k,temp,n,i,j;

int W[10][10];

int C[10][10];

int R[10][10];

int id[10];

int p[10];

int q[10];

void main(){

scanf("%d",&n);

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

scanf("%d",&p[i]);

}

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

scanf("%d",&q[i]);

}

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

W[i][i]=q[i];

C[i][i]=0;

R[i][i]=0;

}

for(int l=1; l<n+1; l++){

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

j = i+l;

W[i][j] = p[j]+q[j]+W[i][j-1];

C[i][j] = 99999;

for(int a=i+1; a<=j; a++){

temp = C[i][a-1]+C[a][j]+W[i][j];

if(temp<C[i][j]){

C[i][j] = temp;

R[i][j]=a;

}

}

}

}

printf("Optimal BST is :: w values are:\n");

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

printf("%d\t",W[i][i]);

}

printf("\n");

for(int l=1; l<n+1; l++){

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

j=i+l;

printf("%d\t",W[i][j]);

}

printf("\n");

}

printf("c values are:\n");

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

printf("%d\t",C[i][i]);

}

printf("\n");

for(int l=1; l<n+1; l++){

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

j=i+l;

printf("%d\t",C[i][j]);

}

printf("\n");

}

printf("r values are:\n");

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

printf("%d\t",R[i][i]);

}

printf("\n");

for(int l=1; l<n+1; l++){

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

j=i+l;

printf("%d\t",R[i][j]);

}

printf("\n");

}

printf("Root is %d\n",R[0][n]);

print(0,n);

}

void print(int i,int j)

{

if(i<j)

{

int t=R[i][j];

printf("Left child of %d is :: %d\n",R[i][j],R[i][t-1]);

printf("Right child of %d is :: %d\n",R[i][j],R[t][j]);

print(i,t-1);

print(t,j);

}

}

10)matrix chain multi problem using dynamic prog

#include<stdio.h>

#define z 10

int P[z+1],M[z+1][z+1],S[z+1][z+1];

void print(int i,int j)

{

if(i==j)

printf("a%d",i);

else

{

printf("(");

print(i,S[i][j]);

print(S[i][j]+1,j);

printf(")");

}

}

void main(){

int n,i,j,l,k,temp;

scanf("%d",&n);

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

scanf("%d",&P[i]);

for(j=0; j<n+1; j++){

M[i][j]=0;

S[i][j]=0;

}

}

for(l=1; l<n+1; l++){

for(i=1; i<n-l+1; i++){

j = i+l;

M[i][j] = 99999;

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

temp = M[i][k] + M[k+1][j] + P[i-1]\*P[k]\*P[j];

if(temp < M[i][j]){

M[i][j] = temp;

S[i][j] = k;

}

}

}

}

printf("cost matrix:\n");

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

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

printf("%d\t",M[i][j]);

}

printf("\n");

}

printf("sequence matrix:\n");

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

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

printf("%d\t",S[i][j]);

}

printf("\n");

}

printf("Optimal ordering is: ");

print(1,n);

printf("\ncost of matrix chain multiplication: %d",M[1][n]);

}

11)state n-queens problem to solve the same using backtracking approach

#include<stdio.h>

char a[10][10];

int n;

int l=0;

int check(int row,int col)

{

int i,j;

for(i=0;i<row;i++)

if(a[i][col]=='Q')

return 0;

for(i=row,j=col;i>=0 && j>=0;i--,j--)

if(a[i][j]=='Q')

return 0;

for(i=row,j=col;i>=0 && j<n;i--,j++)

if(a[i][j]=='Q')

return 0;

return 1;

}

void reset()

{

int i,j;

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

for(j=0;j<n;j++)

a[i][j]='+';

}

}

void print()

{

int i,j;

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

for(j=0;j<n;j++)

printf("%c",a[i][j]);

printf("\n\n");

}

}

void backtrack(int row)

{

int i;

if(row==n)

{

l++;

printf("Solution:%d\n\n",l);

print();

return;

}

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

{

if(check(row,i)==1){

a[row][i]='Q';

backtrack(row+1);

a[row][i]='+';

}

}

}

int main()

{

scanf("%d",&n);

reset();

backtrack(0);

printf("Total sol=%d",l);

return 0;

}

12)write a program to solve knapsack problem using brach and bound

#include<stdio.h>

int weight[10],profit[10],n,m;

float U;

int a[10],top=0;

int upper()

{

int i=0,sum=0,profit1=0;

while(i<n)

{

if(a[i]!=0){

if(sum+weight[i]<=m){

sum=sum+weight[i];

profit1+=profit[i];

}

else

{

break;

}

}

i++;

}

return -1\*profit1;

}

double lower()

{

int i=0,sum=0;

float profit1=0;

while(i<n)

{

if(a[i]!=0)

if(sum+weight[i]<=m){

sum=sum+weight[i];

profit1+=profit[i];

}

else

{

profit1=profit1+((double)(profit[i]\*(m-sum))/(double)weight[i]);

break;

}

i++;

}

return -1\*profit1;

}

void print()

{

int i;

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

{

int val1=0,val2=0,val3=0,val4=0;

val1=upper();

val2=lower();

a[i]=0;

val3=upper();

val4=lower();

a[i]=1;

printf("lwr=%d upr=%d\n\n",val2,val1);

printf("lwr1=%d upr1=%d\n\n",val4,val3);

}

}

void backtrack()

{

int i,j,sum=0;

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

{

int val1=0,val2=0,val3=0,val4=0;

a[i]=1;

val1=upper();

val2=lower();

a[i]=0;

val3=upper();

val4=lower();

for(j=0;j<i;j++)

{

sum=sum+(weight[j]\*a[j]);

}

if(val1<=U && val3<=U)

{

if(val2>val4)

a[i]=0;

else

a[i]=1;

}

else if(val1>U)

a[i]=0;

else if(val3>U)

a[i]=1;

if(sum+weight[i]>m)

a[i]=0;

}

}

int main()

{

int i;

scanf("%d",&n);

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

scanf("%d",&weight[i]);

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

scanf("%d",&profit[i]);

scanf("%d",&m);

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

a[i]=1;

U=upper();

printf("\nupper bound =%d\n\nlower bound =%f\n\n",upper(),lower());

print();

backtrack();

printf("solution is\n\n");

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

printf("%d ",a[i]);

int sum=0;

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

if(a[i]!=0)

sum+=profit[i];

printf("\n\nProfit is %d",sum);

}

13)state the sum of subsets problem & write the program using backtracking.

#include<stdio.h>

#include<stdlib.h>

int s[20],a[20],n,m,k=-1;

void subset(int sum,int top,int mode,int i)

{

i++;

if(i!=n)

{

if(mode==1){

sum=sum+a[i];

s[++top]=a[i];

}

if(sum==m){

for(int j=0;j<=top;j++){

printf("%d ",s[j]);

}

printf("\n");

k=1;

}

else if(sum<m){

subset(sum,top,1,i);

subset(sum,top,0,i);

}

}

}

void main()

{

scanf("%d%d",&n,&m);

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

{

scanf("%d",&a[i]);

}

subset(0,-1,0,-1);

subset(0,-1,1,-1);

if(k==-1)

{

printf("No Solution");

}

}