|  |  |  |
| --- | --- | --- |
| #include<stdio.h> #binomial coefficient  void main()  {int i,j,k,n,c[50][50]; printf("\n enter the value of n & k\n"); scanf("%d%d",&n,&k); for(i=0;i<=n;i++)  for(j=0;j<=k;j++) c[i][j]=0; for(i=0;i<=n;i++) {  c[i][0]=1; c[i][i]=1; } for(i=2;i<=n;i++)  for(j=1;j<=i-1;j++) c[i][j]=c[i-1][j-1]+c[i-1][j];  printf("\n the table for valuation is\n");  for(i=0;i<=n;i++) { for(j=0;j<=k;j++)  if(c[i][j]!=0) printf("\t%d",c[i][j]); printf("\n"); }  printf("\n\t the binomial coefficient of C(%d,%d) is %d\n",n,k,c[n][k]);  }  #include<stdio.h> #breadth first search  void distance(int,int);  int a[10][10];  void main()  {int i,j,n; printf("\n Enter the number of vertices in the diagraph:");scanf("%d",&n);  printf("\n Enter the adjacency matrix\n");  for(i=1;i<=n;i++) for(j=1;j<=n;j++)  scanf("%d",&a[i][j]); for(i=1;i<=n;i++)  {printf("\n\t the starting vertex is %d\n",i);  distance(i,n);  printf("\n \t press enter for other source vertex\n");  }} void distance(int v,int n)  { int queue[40],visited[20],dis[20],front,rear,i,j;  for(i=1;i<=n;i++) visited[i]=dis[i]=0;  front=rear=0; queue[rear++]=v;  visited[v]=1; do { i=queue[front++];  for(j=1;j<=n;j++) if(a[i][j] && !visited[j]) {  dis[j]=dis[i]+1; queue[rear++]=j; visited[j]=1;  printf("\n\t the vertex %d to %d is of distance=%d\n",v,j,dis[j]); } } while(front<rear); }  #include<stdio.h> #depth first search  void dfs(int n,int cost[10][10],int u,int s[])  { int v; s[u]=1; for(v=0;v<n;v++) {  if(cost[u][v]==1 && s[v]==0) { dfs(n,cost,v,s); } } }  void main() {  int n,i,j,cost[10][10],s[10],connected,flag;  printf("\n enter the number of nodes\n");  scanf("%d",&n); printf("\n enter the adjacency matrix\n"); for(i=0;i<n;i++) { for(j=0;j<n;j++)  { scanf("%d",&cost[i][j]); } } connected=0;  for(j=0;j<n;j++) { for(i=0;i<n;i++)  s[i]=0; dfs(n,cost,j,s); flag=0; for(i=0;i<n;i++)  {if(s[i]==0) flag=1; } if(flag==0) connected=1; }  if(connected==1) printf("graph is connected\n");  else printf("graph is not connected\n"); }  #include<stdio.h> #djsk  int main ()  {  int n, cost[15][15], i, j, s[15], v, u, w, dist[15], num, min;  printf ("Enter the vertices please\n");  scanf ("%d", &n);  printf ("Enter the cost of the edges please\n");  printf ("Enter 999 if the edgeis not present or for the self loop\n"); for (i = 1; i <= n; i++)  for (j = 1; j <= n; j++) scanf ("%d", &cost[i][j]);  printf ("Enter the Source vertex please\n");  scanf ("%d", &v); for (i = 1; i <= n; i++) {  s[i] = 0; dist[i] = cost[v][i]; } s[v] = 1;  dist[v] = 0; for (num = 2; num <= n - 1; num++)  { min = 999; for (w = 1; w <= n; w++)  if (s[w] == 0 && dist[w] < min) { min = dist[w];  u = w; }  s[u] = 1;  for (w = 1; w <= n; w++) { if (s[w] == 0)  {  if (dist[w] > (dist[u] + cost[u][w])) dist[w] = (dist[u] + cost[u][w]); } } }  printf ("VERTEX\tDESTINATION\tCOST\n");  for (i = 1; i <= n; i++)  printf (" %d\t %d\t\t %d\n", v, i, dist[i]);  } | #include<stdio.h> #include<stdlib.h>  #include<string.h> #horspools  void main() { int table[126]; char t[100],p[25];  int n,i,k,j,m,flag=0; printf("Enter the Text\n");  gets(t); n=strlen(t); printf("Enter the Pattern\n");  gets(p); m=strlen(p); for(i=0;i<126;i++) table[i]=m;  for(j=0;j<=m-2;j++) table[p[j]]=m-1-j; i=m-1;  while(i<=n-1) { k=0;  while(k<=m-1 && p[m-1-k] == t[i-k])  k++; if(k == m) {  printf("The position of the pattern is %d\n",i-m+2);  flag=1; break; } Else i=i+table[t[i]];} if(!flag)  printf("Pattern is not found in the given text\n"); }  #include<stdio.h> #floyds algo  #include<stdlib.h>  int cost[10][10],a[10][10];  void all\_paths(int [10][10],int [10][10],int);  int min1(int,int); void main() {  int i,j,n; printf("\n enter the number of vertices\n");  scanf("%d",&n); printf("\n enter the adjacency matrix\n"); for(i=1;i<=n;i++) for(j=1;j<=n;j++)  scanf("%d",&cost[i][j]); all\_paths(cost,a,n);  printf("\n\t the shortest path obtained is\n");  for(i=1;i<=n;i++) {  for(j=1;j<=n;j++)printf("\t %d",a[i][j]);printf("\n"); } }  void all\_paths(int cost[10][10],int a[10][10],int n)  { int i,j,k; for(i=1;i<=n;i++) for(j=1;j<=n;j++)  a[i][j]=cost[i][j]; for(k=1;k<=n;k++) for(i=1;i<=n;i++)  for(j=1;j<=n;j++) a[i][j]=min1(a[i][j],a[i][k]+a[k][j]); }  int min1(int a,int b) { return(a<b)?a:b; }  #include<stdio.h> #warshells  int a[10][10]; void main() { int i,j,k,n;  printf("\n enter the number of vertices\n");  scanf("%d",&n); printf("\n enter the adjacency matrix\n"); for(i=1;i<=n;i++) for(j=1;j<=n;j++)  scanf("%d",&a[i][j]); for(k=1;k<=n;k++)  for(i=1;i<=n;i++) for(j=1;j<=n;j++)  a[i][j]=a[i][j] || a[i][k] && a[k][j];  printf("\n\t the tranitive closure is\n");  for(i=1;i<=n;i++) { for(j=1;j<=n;j++)  printf("\t %d",a[i][j]); printf("\n"); } }  #include<stdio.h> #kruskals  int root[10], flag = 0, count=0, temp, min;  int a[20], cost[20][20], n, i, j, k, totalcost = 0, x, y;  void find\_min (), check\_cycle (), update ();  int main () { printf ("Enter the number of vertices please\n"); scanf ("%d", &n); printf ("Enter the cost of the matrix please\n"); for (i = 1; i <= n; i++) for (j = 1; j <= n; j++) scanf ("%d", &cost[i][j]); find\_min ();  while (min != 999 && count != n - 1) { check\_cycle (); if (flag) { printf ("%d ---> %d = %d\n", x, y, cost[x][y]); totalcost += cost[x][y]; update ();  count++; } cost[x][y] = cost[y][x] = 999;  find\_min (); } if (count < n - 2) printf ("The graph is not connected\n"); else  printf ("The graph is connected & the min cost is %d\n", totalcost); } void check\_cycle ()  {  if ((root[x] == root[y]) && (root[x] != 0)) flag = 0;  else flag = 1;} void find\_min () {  min = 999; for (i = 1; i <= n; i++) for (j = 1; j <= n; j++)  if (min > cost[i][j]) { min = cost[i][j];  x = i; y = j; } } void update ()  { if (root[x] == 0 && root[y] == 0) root[x] = root[y] = x;  else if (root[x] == 0)  root[x] = root[y]; else if (root[y] == 0)  root[y] = root[x];  else{ temp = root[y];  for (i = 1; i <= n; i++) if (root[i] == temp)  root[i] = root[x]; } } | #include<stdio.h> #include<stdlib.h> #knapsack  int v[20][20]; int max1(int a,int b)  { return(a>b)?a:b; } void main()  { int i,j,p[20],w[20],n,max;  printf("\n enter the number of items\n");  scanf("%d",&n); for(i=1;i<=n;i++) {  printf("\n enter the weight and profit of the item %d:",i);  scanf("%d %d",&w[i],&p[i]); }  printf("\n enter the capacity of the knapsack");  scanf("%d",&max); for(i=0;i<=n;i++)  v[i][0]=0; for(j=0;j<=max;j++) v[0][j]=0;  for(i=1;i<=n;i++) for(j=1;j<=max;j++) {  if(w[i]>j) v[i][j]=v[i-1][j];  else v[i][j]=max1(v[i-1][j],v[i-1][j-w[i]]+p[i]); }  printf("\n\nThe table is\n");  for(i=0;i<=n;i++) { for(j=0;j<=max;j++)  printf("%d\t",v[i][j]); printf("\n"); }  printf("\nThe maximum profit is %d",v[n][max]);  printf("\nThe most valuable subset is:{");  j=max; for(i=n;i>=1;i--) if(v[i][j]!=v[i-1][j]) {  printf("\t item %d:",i); j=j-w[i]; } printf("}"); }  #include<stdio.h> void main() #prims algo  { int cost[20][20],t[20][20],near1[20],a[20];  int i,j,n,min,minimum,k,l,mincost,c,b;  printf("\n enter the number of nodes\n");  scanf("%d",&n);  printf("\n enter the adjacency matrix\n");  for(i=1;i<=n;i++) for(j=1;j<=n;j++)  scanf("%d",&cost[i][j]); minimum=cost[1][1];  for(i=1;i<=n;i++) for(j=1;j<=n;j++) {  if(minimum>=cost[i][j]) { minimum=cost[i][j];  k=i; l=j; } } mincost=minimum; t[1][1]=k;  t[1][2]=l; for(i=1;i<=n;i++) { if(cost[i][l]<cost[i][k])  near1[i]=l; else near1[i]=k; } near1[k]=near1[l]=0;  for(i=2;i<=n-1;i++) { min=999; for(j=1;j<=n;j++) {  if(near1[j]!=0) { a[j]=cost[j][near1[j]]; { min=a[j];  c=near1[j]; b=j; printf("\n"); } } }  mincost=mincost+cost[b][c]; near1[b]=0;  for(k=1;k<=n;k++) if((near1[k]!=0) && (cost[k][near1[k]]>cost[k][b])) near1[k]=b; }  printf("\n the cost of minimum spanning tree is=%d",mincost); }  #include<stdio.h> #include<math.h> #n queens  int x[20],count=1; void queens(int,int);  int place(int,int); void main() {  int n,k=1;  printf("\n enter the number of queens to be placed\n");  scanf("%d",&n); queens(k,n); }  void queens(int k,int n) { int i,j; for(j=1;j<=n;j++)  { if(place(k,j)) { x[k]=j; if(k==n) {  printf("\n %d solution",count); count++; for(i=1;i<=n;i++)  printf("\n \t %d row <---> %d column",i,x[i]); }  else  queens(k+1,n); } } } int place(int k,int j) {  int i; for(i=1;i<k;i++) if((x[i]==j) || (abs(x[i]-j))==abs(i-k))  return 0; return 1; }  #include<stdio.h> #define max 20 #topological  #include<stdlib.h> int a[max][max],n;  void topological\_sort(); void main()  { int i,j; printf("\n enter the number of vertices\n");  scanf("%d",&n);  printf("\n enter the adjacency matrix\n");  for(i=1;i<=n;i++) for(j=1;j<=n;j++)  scanf("%d",&a[i][j]); topological\_sort(); }  void topological\_sort() {  int v[max],ver[max],i,j,p=1,flag=0; for(i=1;i<=n;i++)  v[i]=0; while(p<=n) { j=1; while(j<=n) {  flag=0; if(v[j]==0) { for(i=1;i<=n;i++)  if((a[i][j]!=0) && (v[i]==0)) { flag=1; break; }  if(flag==0) { v[j]=1; ver[p++]=j; break; } }  j++; if(j>n) {  printf("\n topological order is not possible\n");  exit(0); } } }  printf("\n topological order obtained is...\n");  for(i=1;i<p;i++) printf("\t%d",ver[i]); } |

|  |  |  |
| --- | --- | --- |
| //heap sort  #include<stdio.h>#include<stdlib.h>#include<time.h>  void heap(int a[],int n) {int i,j,k,temp; for(i=2;i<=n;i++) { j=i; k=j/2; temp=a[j]; while(k>0 && a[k]<temp)  { a[j]=a[k]; j=k; k=k/2; } a[j]=temp; } }  void heap1(int a[],int n) { int i,j,k,temp; for(i=n/2;i>0;i--)  { k=i; temp=a[k]; j=2\*k; while(j<=n) {  if(j<n && a[j]<a[j+1]) { j=j+1; } if(temp<a[j]) { a[k]=a[j]; k=j; j=2\*k; } else { break; } } a[k]=temp; } } void adjust(int a[],int n) { int i=2,temp=a[1]; while(i<=n) { if(i<n && a[i]<a[i+1]) { i=i+1; } if(a[i]>temp) { a[i/2]=a[i]; i=i\*2; } else { break; } } a[i/2]=temp; } void main() {  int a[10000],n,i,temp,choice; clock\_t st,et; system("clear"); do { printf("\nEnter the value of n:\n");  scanf("%d",&n); for(i=1;i<=n;i++) { a[i]=n-i; }  printf("\nThe elements of the array are:\n");  for(i=1;i<n;i++) printf("%d ",a[i]); st=clock(); heap1(a,n);  for(i=n;i>=2;i--) { temp=a[1]; a[1]=a[i]; a[i]=temp;  adjust(a,i-1); }et=clock(); printf("\nThe sorted elements are:\n"); for(i=1;i<=n;i++) printf("%d ",a[i]);  double t=(et-st);  printf("\nTime taken is %lf",t/CLOCKS\_PER\_SEC);  printf("\nDo you want to continue press 1 else 0\n");  scanf("%d",&choice); } while(choice); }  #include<stdio.h>#include<time.h> #mergesort  #include<stdlib.h>#define MAX 2000  void mergesort(int[],int,int);void merge(int[],int,int,int);  void main(){ int ch=1; double t; int n,i,a[MAX],low,high;  clock\_t begin,end; system("clear"); srand(time(NULL));  while(ch) { printf("\nEnter the no. of ele\n");  scanf("%d",&n); for(i=0;i<n;i++) a[i]=rand()%MAX;  printf("\nThe elements of the array randamly generated are\n"); for(i=0;i<n;i++) printf("%d ", a[i]);  low=0;high=n-1; begin=clock(); mergesort(a,low,high); end=clock(); printf("\nSorted elements after applying merge sort\n"); for(i=0;i<n;i++)  printf("%d ",a[i]); double t=end-begin; printf("\nTime taken is %lf seconds\n",t/CLOCKS\_PER\_SEC);  printf("\nDo you wish to run again(1/0)\n");  scanf("%d",&ch); } }  void mergesort(int a[],int low,int high) {  int mid; if(low<high) { mid=(low+high)/2;  mergesort(a,low,mid); mergesort(a,mid+1,high);  merge(a,low,mid,high); } }  void merge(int a[],int low,int mid,int high) {  int i,j,k,t[MAX]; i=low; j=mid+1; k=low;  while((i<=mid)&&(j<=high)) if(a[i]<=a[j]) t[k++]=a[i++];  else t[k++]=a[j++]; while(i<=mid) t[k++]=a[i++];  while(j<=high) t[k++]=a[j++]; for(i=low;i<=high;i++)  a[i]=t[i]; } | #include<stdio.h>#include<time.h> #linear binary  #include<stdlib.h> #define MAX 2000  int pos; int binsearch(int,int[],int,int,int);  int linsearch(int,int[],int); void mergesort(int [], int, int);  void merge(int [], int, int, int); void main() { int ch=1;  int n,i,a[MAX],k,op,low,high,pos; clock\_t begin,end;  srand(time(NULL)); while(ch) {  printf("\n.....MENU.....\n 1.Binary Search\n 2.Linear Search\n 3.Exit\n"); printf("\nEnter your choice\n");  scanf("%d",&op); switch(op) {  case 1:printf("\nEnter the number of elements \n"); scanf("%d",&n); for(i=0;i<n;i++) a[i] = rand()%MAX; printf("The Elements of the array randomly generated are \n"); for(i=0;i<n;i++) printf("%d ",a[i]); low=0;high=n-1; mergesort(a, low, high); printf("\nsorted elements after applying the mergesort\n"); for(i=0;i<n;i++) printf("%d ",a[i]); printf("\nEnter the element to be searched\n"); scanf("%d",&k); begin=clock(); pos=binsearch(n,a,k,low,high); end=clock(); if(pos==-1) printf("\n\n Unsuccessful search"); else printf("\n Element %d is found at position %d",k,pos+1); printf("\n Time taken is %f CPU1 cycles\n",((double)end-begin)/CLOCKS\_PER\_SEC);  break;  case 2:printf("\nEnter the number of elements\n"); scanf("%d",&n); for(i=0;i<n;i++)a[i] = rand()%MAX; printf("The Elements of the array randomly generated are \n"); for(i=0;i<n;i++)printf("%d ",a[i]);  printf("\nEnter the element to be searched\n"); scanf("%d",&k); begin=clock(); pos=linsearch(n,a,k); end=clock(); if(pos==-1) printf("\n\n Unsuccessful search"); printf("\n Element %d is found at position %d",k,pos+1);  printf("\n Time taken is %f CPU cycles\n",((double)end-begin)/CLOCKS\_PER\_SEC);  break; default:printf("\nInvalid choice entered\n"); exit(0); }  printf("\n Do you wish to run again (1/0) \n"); scanf("%d",&ch); } }  int binsearch(int n,int a[],int k,int low,int high){  int mid;  mid=(low+high)/2; if(low>high) return -1; if(k==a[mid]) return(mid); else if(k<a[mid]) return binsearch(n,a,k,low,mid-1); else return binsearch(n,a,k,mid+1,high); } int linsearch(int n,int a[],int k) {  if(n<0) return -1; if(k==a[n-1]) return(n-1);  else return linsearch(n-1,a,k); }void mergesort(int a[],int low,int high) { int mid;  if(low<high){mid=(low+high)/2; mergesort(a,low,mid); mergesort(a,mid+1,high);  merge(a,low,mid,high); } }  void merge(int a[],int low,int mid,int high) {  int i,j,k,t[MAX]; i=low; j=mid+1; k=low;  while((i<=mid) && (j<=high) if(a[i]<=a[j])  t[k++]=a[i++]; else t[k++]=a[j++]; while(i<=mid)  t[k++]=a[i++]; while(j<=high) t[k++]=a[j++];  for(i=low;i<=high;i++) a[i]=t[i]; } | #include<stdio.h>#include<stdlib.h> #quick sort  #include<time.h>#define MAX 2000  void quicksort(int[],int,int);int partition(int[],int,int);  void main(){ int i,n,a[MAX],ch=1, choice; double t; clock\_t st, et; system("clear"); while(ch) { printf("\n 1: Worst case analysis \n 2: Average Case Analysis \n"); printf("\n Enter your choice\n"); scanf("%d", &choice); switch(choice) { case 1: printf("\n \*\*\*\*\*Analysis of worst case time complexity of quick is T(n)=bigO(n^2)\*\*\*\* \n"); printf("\n \*\*\*\*\* And when all the elements are in sorted order\*\*\*\*\*\*\n"); printf("\nEnter the number of elements\n"); scanf("%d",&n); for(i=0;i<n;i++) a[i] = i+1; printf("The Elements considered for the worst case analysis are \n"); for(i=0;i<n;i++) printf("%d ",a[i]); st = clock(); quicksort(a,0,n-1); et = clock(); printf("\n\nthe sorted array elements are\n\n"); for(i=0;i<n;i++) printf("%d ",a[i]); t=et-st; printf("\n The time taken is %f \n",t/CLOCKS\_PER\_SEC);  break; case 2: printf("\n \*\*\*\*\*Analysis of Average case time complexity of quick is T(n)=1.38\*nlog2(n) Note: 2 indicates base 2 \*\*\*\* \n"); printf("\n \*\*\* And when all the elements are randomly considered\*\*\*\n"); printf("\nEnter the number of elements\n"); scanf("%d",&n); for(i=0;i<n;i++) a[i] = rand()%MAX; printf("The Elements considered for the Average case analysis are \n"); for(i=0;i<n;i++) printf("%d ",a[i]); st = clock(); quicksort(a,0,n-1); et = clock(); printf("\n\nThe sorted array elements are\n\n"); for(i=0;i<n;i++)printf("%d ",a[i]); t=et-st; printf("\nThe time taken is %f \n",t/CLOCKS\_PER\_SEC);  break; case 3: printf("\nInvalid Choice \n");  break; } printf("\n\nDo u wish to continue (0/1)\n"); scanf("%d",&ch);  } }  void quicksort(int a[],int low,int high) { int mid;if(low<high) {mid=partition(a,low,high);quicksort(a,low,mid-1);quicksort(a,mid+1,high); } } int partition(int a[],int low,int high) { int key,i,j,temp,k; key=a[low]; i=low+1; j=high; while(i<=j) { while(i<=high && key>=a[i]) i=i+1; while(key<a[j]) j=j-1; if(i<j) {temp=a[i];a[i]=a[j]; a[j]=temp; }else {k=a[j]; a[j]=a[low];a[low]=k; }  } return j; } |