Max Heap- Deletion

```
#include<stdio.h>

void maxheap(int a[],int n);

void heapfy(int a[],int n,int i);

void deleteNode(int a[],int *n,int value);

void main(){

 int a[20],n,i,c=0,val;

 printf("Enter size: "); scanf("%d",&n);

 printf("Enter elements: ");

 for(i=1;i<=n;i++){ scanf("%d",&a[i]); maxheap(a,++c);}

 printf("Heap: "); for(i=1;i<=n;i++) printf("%d ",a[i]);

 printf("\nEnter value to delete: "); scanf("%d",&val);

 deleteNode(a,&n,val);

 printf("After deletion: "); for(i=1;i<=n;i++) printf("%d ",a[i]);

}

void maxheap(int a[],int n){ int i; for(i=n/2;i>0;i--) heapfy(a,n,i);}

void heapfy(int a[],int n,int i){

 int temp,large=i,left=2*i,right=2*i+1;

 if(left<=n&&a[left]>a[large]) large=left;

 if(right<=n&&a[right]>a[large]) large=right;

 if(i!=large){ temp=a[large]; a[large]=a[i]; a[i]=temp; heapfy(a,n,large);}

}

void deleteNode(int a[],int *n,int value){

 int i,pos=-1;

 for(i=1;i<=*n;i++) if(a[i]==value){ pos=i; break; }

 if(pos==-1){ printf("Value not found!\n"); return;}

 a[pos]=a[(*n)--]; maxheap(a,*n);

}
```

```
Enter size: 5
Enter elements: 5
4
3
2
1
Heap: 5 4 3 2 1
Enter value to delete: 2
After deletion: 5 4 3 1
```

MINHEAP-DELETION

```
#include<stdio.h>
void minheap(int a[],int n);
void heapfy(int a[],int n,int i);
void deleteNode(int a[],int *n,int value);
void main(){
 int a[20],n,i,c=0,val;
 printf("Enter size: "); scanf("%d",&n);
 printf("Enter elements: ");
 for(i=1;i<=n;i++){ scanf("%d",&a[i]); minheap(a,++c);}
 printf("Heap: "); for(i=1;i<=n;i++) printf("%d ",a[i]);
 printf("\nEnter value to delete: "); scanf("%d",&val);
 deleteNode(a,&n,val);
 printf("After deletion: "); for(i=1;i<=n;i++) printf("%d ",a[i]);
}
void minheap(int a[],int n){ int i; for(i=n/2;i>0;i--) heapfy(a,n,i);}
void heapfy(int a[],int n,int i){
 int temp,small=i,left=2*i,right=2*i+1;
 if(left<=n&&a[left]<a[small]) small=left;
 if(right<=n&&a[right]<a[small]) small=right;
 if(i!=small){ temp=a[small]; a[small]=a[i]; a[i]=temp; heapfy(a,n,small);}
}
void deleteNode(int a[],int *n,int value){
 int i,pos=-1;
 for(i=1;i<=*n;i++) if(a[i]==value){ pos=i; break; }
 if(pos==-1){ printf("Value not found!\n"); return;}
 a[pos]=a[(*n)--]; minheap(a,*n);
}
```

```
Enter size: 5
Enter elements: 5
4
3
2
1
Heap: 1 2 4 5 3
Enter value to delete: 4
After deletion: 1 2 3 5
```

HEAP-SORT

```
#include<stdio.h>

void heapsort(int a[],int n); void heapify(int a[],int n,int i);

void main(){

 int a[20],n,i;

 printf("Enter size: "); scanf("%d",&n);

 printf("Enter elements: "); for(i=1;i<=n;i++) scanf("%d",&a[i]);

 printf("Before sorting: "); for(i=1;i<=n;i++) printf("%d ",a[i]);

 heapsort(a,n);

 printf("\nAfter sorting: "); for(i=1;i<=n;i++) printf("%d ",a[i]);

}

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

 int i,temp;

 for(i=n/2;i>0;i--) heapify(a,n,i);

 for(i=n;i>0;i--){

  temp=a[1]; a[1]=a[i]; a[i]=temp;

  heapify(a,i-1,1);

 }

}

void heapify(int a[],int n,int i){

 int large=i,left=2*i,right=2*i+1,temp;

 if(left<=n&&a[left]>a[large]) large=left;

 if(right<=n&&a[right]>a[large]) large=right;

 if(large!=i){ temp=a[large]; a[large]=a[i]; a[i]=temp; heapify(a,n,large);;}

}
```

```
Enter size: 5
Enter elements: 5
2
3
4
1
Before sorting: 5 2 3 4 1
After sorting: 1 2 3 4 5
```

BFS TRAVERSAL

```
#include<stdio.h>

void bfs(int v);

int a[20][20],queue[20],visited[20],n,front=-1,rear=-
1;

void main(){

 int v,i,j;

 printf("Enter number of vertices: "); scanf("%d",&n);  printf("Enter 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++) visited[i]=0;

 printf("Enter starting vertex: "); scanf("%d",&v);     front=rear=0; queue[rear]=v; visited[v]=1;
printf("BFS Traversal: %d",v);     bfs(v); }   void bfs(int v){

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

  if(a[v][i]!=0&&visited[i]==0){

   queue[++rear]=i; visited[i]=1; printf("%d",i); }  }

 if(++front<=rear) bfs(queue[front]); }
```

```
Enter number of vertices: 5
Enter adjacency matrix:01110
10011
10010
11101
01010
Enter starting vertex: 3
BFS Traversal: 3 1 4 2 5
```

DFS-TRAVERSAL

```
#include<stdio.h>

void dfs(int v);

int a[20][20],visited[20],n;

void main(){

 int v,i,j;

 printf("Enter number of vertices: "); scanf("%d",&n);

 printf("Enter 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++) visited[i]=0;    printf("Enter starting vertex: "); scanf("%d",&v);

 printf("DFS Traversal: ");  dfs(v);  }

void dfs(int v) {  int i;   printf("%d ",v); visited[v]=1;

 for(i=1;i<=n;i++)   if(a[v][i]!=0&&visited[i]==0) dfs(i);  }
```

```
Enter number of vertices: 5
Enter adjacency matrix:01110
10011
10010
11101
01010
Enter starting vertex: 3
BFS Traversal: 3 1 2 4 5
```

CONNECTED COMPONENTS

```
#include<stdio.h>

int a[20][20],visited[20],n;   void dfs(int v); int connected();

void main(){    int i,u,v,e,c;

 printf("Enter vertices & edges: "); scanf("%d%d",&n,&e);
for(i=1;i<=e;i++){

  printf("Enter edge (u v): ");   scanf("%d%d",&u,&v);    a[u][v]=a[v][u]=1;

 }  c=connected();    printf("Connected Components: %d",c); }

int connected(){     int i,count=0;   for(i=1;i<=n;i++) visited[i]=0;

 for(i=1;i<=n;i++) if(!visited[i]){ dfs(i); count++; }   return count; }

void dfs(int v){    int i; visited[v]=1;    for(i=1;i<=n;i++) if(a[v][i]&&!visited[i]) dfs(i);  }
```

```
Enter vertices & edges: 5 5
Enter edge (u v): 1 2
Enter edge (u v): 2 3
Enter edge (u v): 3 1
Enter edge (u v): 4 5
Enter edge (u v): 5 4
Connected Components: 2
```

QUICK-SORT

```
#include<stdio.h>

void quicksort(int a[],int low,int high);

int partition(int a[],int low,int high);     void main(){     int a[20],n,i;

 printf("Enter size: "); scanf("%d",&n);

 printf("Enter elements: "); for(i=0;i<n;i++) scanf("%d",&a[i]);

 printf("Before sorting:\n"); for(i=0;i<n;i++) printf("%d ",a[i]);

 quicksort(a,0,n-1);

 printf("\nAfter sorting:\n"); for(i=0;i<n;i++) printf("%d ",a[i]);  }

int partition(int a[],int low,int high){

 int i=low,j=high,temp,pivot=a[low];     while(i<j){      while(a[i]<=pivot) i++;

  while(a[j]>pivot) j--;     if(i<j){ temp=a[i]; a[i]=a[j]; a[j]=temp; }   }

 a[low]=a[j]; a[j]=pivot;    return j;   }  void quicksort(int a[],int low,int high){

 if(low<high){      int loc=partition(a,low,high);    quicksort(a,low,loc-1);
quicksort(a,loc+1,high);  } }
```

```
Enter size: 5
Enter elements: 10
30
50
40
20
Before sorting:
10 30 50 40 20
After sorting:
10 20 30 40 50
```

MERGE SORT

```
#include<stdio.h>

void mergesort(int a[],int low,int high);

void merge(int a[],int low,int mid,int high);

void main(){   int a[20],n,i;    printf("Enter size: "); scanf("%d",&n);

 printf("Enter elements: "); for(i=0;i<n;i++) scanf("%d",&a[i]);

 printf("Before sorting:\n"); for(i=0;i<n;i++) printf("%d ",a[i]);

 mergesort(a,0,n-1);   printf("\nAfter sorting:\n"); for(i=0;i<n;i++) printf("%d ",a[i]);   }

void mergesort(int a[],int low,int high){     if(low<high){

  int 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=low,j=mid+1,k=low,b[20],x;    while(i<=mid&&j<=high) b[k++]=a[i]<a[j]?a[i++]:a[j++];

 while(i<=mid) b[k++]=a[i++];

 while(j<=high) b[k++]=a[j++];  for(x=low;x<=high;x++) a[x]=b[x];  }
```

```
Enter size: 5
Enter elements: 100
90
80
70
60
Before sorting:
100 90 80 70 60
After sorting:
60 70 80 90 100
```

JOB SEQUENCING

```
#include<stdio.h>  void main(){   int i,j,p[20],d[20],a[20],slot[20],temp,r,total=0,max,n;

 printf("Enter no. of jobs: "); scanf("%d",&n);

 printf("Enter profits: "); for(i=0;i<n;i++) scanf("%d",&p[i]);

 printf("Enter deadlines: "); for(i=0;i<n;i++) scanf("%d",&d[i]);

 max=d[0]; for(i=0;i<n;i++){ a[i]=i; if(d[i]>max) max=d[i]; }

 for(i=0;i<max;i++) slot[i]=-1;    for(i=0;i<n-1;i++) for(j=0;j<n-1-i;j++)

  if(p[j]<p[j+1]){ temp=p[j]; p[j]=p[j+1]; p[j+1]=temp; temp=a[j]; a[j]=a[j+1]; a[j+1]=temp; }

 for(i=0;i<n;i++){ j=a[i]; r=d[j];

while(r>=1){ if(slot[r-1]==-1){ slot[r-1]=j; total+=p[i]; break; } r--; }  }

 printf("Optimal job sequence:\n(");

 for(i=0;i<max;i++){ printf("J%d",slot[i]+1); if(i<max-1) printf(" , "); }

 printf(")\nTotal Profit: %d",total);  }
```

```
Enter no. of jobs: 5
Enter profits: 60 100 20 40
65
Enter deadlines: 2 1 3 2 1
Optimal job sequence:
(J2 , J1 , J3) Total Profit: 180
```

SINGLE-SORCE SP

```
#include<stdio.h>

#define INF 9999

void dijkstra(int[][10],int,int);   void main(){   int n,s,i,j,ad[10][10];

 printf("Enter no. of vertices: "); scanf("%d",&n);   printf("Enter adjacency matrix:\n");

 for(i=1;i<=n;i++) for(j=1;j<=n;j++) scanf("%d",&ad[i][j]);   printf("Enter source vertex: ");
scanf("%d",&s);   dijkstra(ad,n,s);  }  void dijkstra(int ad[][10],int n,int s){

int cost[10][10],visit[10]={0},d[10],min,i,j,u,v,count=1;

 for(i=1;i<=n;i++) for(j=1;j<=n;j++) cost[i][j] = (ad[i][j]==0) ? INF : ad[i][j];

 for(i=1;i<=n;i++) d[i]=cost[s][i];   visit[s]=1; d[s]=0;   while(count<n){

  min=INF;  for(i=1;i<=n;i++) if(!visit[i] && d[i]<min){ min=d[i]; u=i; }

  visit[u]=1;  for(v=1;v<=n;v++) if(!visit[v] && d[u]+cost[u][v]<d[v]) d[v]=d[u]+cost[u][v];

 count++;  }  printf("Vertex\tDistance from Source\n");

 for(i=1;i<=n;i++) printf("%d\t%d\n",i,d[i]);  }
```

0/1 KNAPSACK

```
#include <stdio.h>

int knapsack(int p[],int wt[],int n,int c);  int max(int a,int b){  return (a>b)?a:b; }

void main(){   int n,profit[10],weight[10],capacity,i;

printf("Enter number of objects: "); scanf("%d",&n);

 printf("Enter profits: "); for(i=0;i<n;i++) scanf("%d",&profit[i]);

 printf("Enter weights: "); for(i=0;i<n;i++) scanf("%d",&weight[i]);

 printf("Enter capacity: "); scanf("%d",&capacity);

 printf("Total profit is: %d",knapsack(profit,weight,n,capacity));

} int knapsack(int p[],int wt[],int n,int c){

 int a[10][10],i,w;    for(i=0;i<=n;i++) for(w=0;w<=c;w++)

   a[i][w] = (i==0||w==0) ? 0 : (wt[i-1]>w ? a[i-1][w] : max(a[i-1][w],a[i-1][w-wt[i-1]]+p[i-1]));

 return a[n][c];  }
```

```
Enter number of objects: 4
Enter profits: 2 3 4 1
Enter weights: 3 4 5 6
Enter capacity: 8
Total profit is: 6
```