BINOMIAL HEAP

#include<stdio.h> #include<stdlib.h>

struct node

{

int n;

int degree;

struct node\* parent;

struct node\* child;

struct node\* sibling;

};

struct node\* MAKE\_bin\_HEAP(); int bin\_LINK(struct node\*,struct node\*); struct node\* CREATE\_NODE(int); struct node\* bin\_HEAP\_UNION(struct node\*,struct node\*); struct node\* bin\_HEAP\_INSERT(struct node\*,struct node\*); struct node\* bin\_HEAP\_MERGE(struct node\*,struct node\*); struct node\* bin\_HEAP\_EXTRACT\_MIN(struct node\*); int REVERT\_LIST(struct node\*); int DISPLAY(struct node\*); struct node\* FIND\_NODE(struct node\*,int); int bin\_HEAP\_DECREASE\_KEY(struct node\*,int,int); int bin\_HEAP\_DELETE(struct node\*,int); int count=1;

struct node\* MAKE\_bin\_HEAP()

{

struct node\* np;

np=NULL;

return np;

}

struct node \* H=NULL;struct node \*Hr=NULL;

int bin\_LINK(struct node\* y,struct node\* z)

{

y->parent=z;

y->sibling=z->child;

z->child=y;

z->degree=z->degree+1;

}

struct node\* CREATE\_NODE(int k)

{

struct node\* p;//new node; p=(struct node\*)malloc(sizeof(struct node)); p->n=k; return p;

}

struct node\* bin\_HEAP\_UNION(struct node\* H1,struct node\* H2)

{

struct node\* prev\_x; struct node\* next\_x; struct node\* x;

struct node\* H=MAKE\_bin\_HEAP(); H=bin\_HEAP\_MERGE(H1,H2); if(H==NULL)

return H;

prev\_x=NULL; x=H; next\_x=x->sibling; while(next\_x!=NULL)

{

if((x->degree!=next\_x->degree)||((next\_x->sibling!=NULL)&&(next\_x->sibling)->degree==x->degree))

{

|  |
| --- |
| prev\_x=x; |
| x=next\_x; |

}

else

{

if(x->n<=next\_x->n)

{

x->sibling=next\_x->sibling;

bin\_LINK(next\_x,x);

}

else

{

if(prev\_x==NULL)

H=next\_x;

else

prev\_x->sibling=next\_x;

bin\_LINK(x,next\_x);

x=next\_x;

}

}

next\_x=x->sibling;

}

return H;

}

struct node\* bin\_HEAP\_INSERT(struct node\* H,struct node\* x)

{

struct node\* H1=MAKE\_bin\_HEAP(); x->parent=NULL; x->child=NULL; x->sibling=NULL; x->degree=0;

H1=x;

H=bin\_HEAP\_UNION(H,H1);

return H;

}

struct node\* bin\_HEAP\_MERGE(struct node\* H1,struct node\* H2)

{

struct node\* H=MAKE\_bin\_HEAP(); struct node\* y; struct node\* z; struct node\* a; struct node\* b; y=H1; z=H2; if(y!=NULL)

{

if(z!=NULL&&y->degree<=z->degree)

H=y;

else if(z!=NULL&&y->degree>z->degree)

/\* need some modifications here;the first and the else conditions can be merged together!!!! \*/

H=z;

else

H=y;

}

else

H=z; while(y!=NULL&&z!=NULL)

{

if(y->degree<z->degree)

{

y=y->sibling;

}

else if(y->degree==z->degree)

{

|  |
| --- |
| a=y->sibling; |
| y->sibling=z; |

y=a;

}

else

{

|  |
| --- |
| b=z->sibling; |
| z->sibling=y; |

z=b;

}

}

return H;

}

int DISPLAY(struct node\* H)

{

struct node\* p;

if(H==NULL)

{

printf("\nHEAP EMPTY");

return 0;

}

printf("\nTHE ROOT NODES ARE:-\n");

p=H;

while(p!=NULL)

{

printf("%d",p->n);

if(p->sibling!=NULL)

printf("-->");p=p->sibling;

}

printf("\n");

}

struct node\* bin\_HEAP\_EXTRACT\_MIN(struct node\* H1)

{ int min; struct node\* t=NULL; struct node\* x=H1; struct node \*Hr; struct node\* p;

Hr=NULL; if(x==NULL)

{

printf("\nNOTHING TO EXTRACT");

return x;

}

// int min=x->n; p=x;

while(p->sibling!=NULL)

{

if((p->sibling)->n<min)

{

min=(p->sibling)->n;

t=p;

x=p->sibling;

}

p=p->sibling;

}

if(t==NULL&&x->sibling==NULL)

H1=NULL;

else if(t==NULL)

H1=x->sibling;

else if(t->sibling==NULL)

t=NULL;

else

t->sibling=x->sibling;

if(x->child!=NULL)

{

REVERT\_LIST(x->child);

(x->child)->sibling=NULL;

}

H=bin\_HEAP\_UNION(H1,Hr);

return x;

}

int REVERT\_LIST(struct node\* y)

{

if(y->sibling!=NULL)

{

REVERT\_LIST(y->sibling);

(y->sibling)->sibling=y;

}

else

{

Hr=y;

}

}

struct node\* FIND\_NODE(struct node\* H,int k)

{

struct node\* x=H; struct node\* p=NULL; if(x->n==k)

{

p=x;

return p;

}

if(x->child!=NULL&&p==NULL)

{

p=FIND\_NODE(x->child,k);

}

if(x->sibling!=NULL&&p==NULL)

{

p=FIND\_NODE(x->sibling,k);

}

return p;

}

int bin\_HEAP\_DECREASE\_KEY(struct node\* H,int i,int k)

{

int temp;

struct node\* p;

|  |
| --- |
| struct node\* y; |
| struct node\* z; |

p=FIND\_NODE(H,i);

if(p==NULL)

{

printf("\nINVALID CHOICE OF KEY TO BE REDUCED");

return 0;

}

if(k>p->n)

{

printf("\nSORRY!THE NEW KEY IS GREATER THAN CURRENT ONE");

return 0;

}

p->n=k; y=p; z=p->parent;

while(z!=NULL&&y->n<z->n)

{

temp=y->n;

y->n=z->n;

z->n=temp;

y=z;

z=z->parent;

}

printf("\nKEY REDUCED SUCCESSFULLY!");

}

int bin\_HEAP\_DELETE(struct node\* H,int k)

{

struct node\* np; if(H==NULL)

{

printf("\nHEAP EMPTY");

return 0;

}

bin\_HEAP\_DECREASE\_KEY(H,k,-1000);

np=bin\_HEAP\_EXTRACT\_MIN(H); if(np!=NULL)

printf("\nNODE DELETED SUCCESSFULLY");

}

int main()

{

int i,n,m,l;

struct node\* p;

struct node\* np;

|  |  |
| --- | --- |
| char ch; | printf("\nENTER THE NUMBER OF ELEMENTS:"); |

scanf("%d",&n);

printf("\nENTER THE ELEMENTS:\n");

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

{

scanf("%d",&m);

np=CREATE\_NODE(m);

H=bin\_HEAP\_INSERT(H,np);

}

DISPLAY(H);

do

{

printf("\nMENU:-\n");

printf("\n1)INSERT AN ELEMENT\n2)EXTRACT THE MINIMUM KEY NODE\n3)DECREASE A

NODE KEY\n 4)DELETE A NODE\n5)QUIT\n");

scanf("%d",&l);

switch(l)

{

case 1:do

{

printf("\nENTER THE ELEMENT TO BE INSERTED:");

scanf("%d",&m);

p=CREATE\_NODE(m);

H=bin\_HEAP\_INSERT(H,p);

printf("\nNOW THE HEAP IS:\n");

DISPLAY(H);

printf("\nINSERT MORE(y/Y)= \n");

fflush(stdin);

scanf("%c",&ch);

}while(ch=='Y'||ch=='y');

break;

case 2:do

{

printf("\nEXTRACTING THE MINIMUM KEY NODE");

p=bin\_HEAP\_EXTRACT\_MIN(H);

if(p!=NULL)

printf("\nTHE EXTRACTED NODE IS %d",p->n);

printf("\nNOW THE HEAP IS:\n");

DISPLAY(H);

printf("\nEXTRACT MORE(y/Y)\n");

fflush(stdin);

scanf("%c",&ch);

}while(ch=='Y'||ch=='y');

break;

case 3:do

{

printf("\nENTER THE KEY OF THE NODE TO BE DECREASED:");

scanf("%d",&m);

printf("\nENTER THE NEW KEY : ");

scanf("%d",&l);

bin\_HEAP\_DECREASE\_KEY(H,m,l);

printf("\nNOW THE HEAP IS:\n");

DISPLAY(H);

printf("\nDECREASE MORE(y/Y)\n");

fflush(stdin);

scanf("%c",&ch);

}while(ch=='Y'||ch=='y');

break;

case 4:do

{

printf("\nENTER THE KEY TO BE DELETED: ");

scanf("%d",&m);

bin\_HEAP\_DELETE(H,m);

printf("\nDELETE MORE(y/Y)\n");

fflush(stdin);

scanf("%c",&ch);

}while(ch=='y'||ch=='Y');

break;

case 5:printf("\nTHANK U \n");break;

default :printf("\nINVALID ENTRY...TRY AGAIN....\n");

}

}while(l!=5);

}