Funzioni di libreria

List.h

```
link newNode(Item val, link next);
link listInsHead(link h, Item val);
void listInsHeadP(link *hp, Item val);
link listInsTail(link h, Item val);
void listInsTailP(link *hp, Item val);
void listInsTFast(link *hp, link *tp, Item val);
link SortListIns(link h, Item val);
void listTravR(link h);
void listRevTravR(link h);
Item listSearch(link h, Key k);
Item SortListSearch(link h, Key k);
link listDelHead(link h);
link listDelKey(link h, Key k);
link listDelKeyR(link x, Key k);
link SortListDel(link h, Key k);
Item listExtrHeadP(link *hp);
Item listExtrKeyP(link *hp, Key k);
link listReverseF(link x);
link listSortF(link h);
```

Set.h

```
SET SETinit(int maxN);
void SETfree(SET s);
void SETfill(SET s, Item val);
int SETsearch(SET s, Key k);
SET SETunion(SET s1, SET s2);
SET SETintersection(SET s1, SET s2);
int SETsize(SET s);
```

```
int SETempty(SET s);
void SETdisplay(SET s);
```

Stack.h

```
void STACKinit(int maxN);
int STACKempty();
void STACKpush(Item val);
Item STACKpop();
```

Queue.h

```
QUEUE QUEUEinit(int maxN);
int QUEUEempty(QUEUE q);
void QUEUEput(QUEUE q, Item val);
Item QUEUEget(QUEUE q);
```

PQ.h

```
PQ PQinit(int maxN);
void PQfree(PQ pq);
int PQempty(PQ pq);
void PQinsert(PQ pq, Item data);
Item PQextractMax(PQ pq);
Item PQshowMax(PQ pq);
void PQdisplay(PQ pq);
void PQchange(PQ pq, Item data);
```

UF.h

```
int UFfind(int p, int q);
void UFunion(int p, int q);
```

Heap.h

```
Heap HEAPinit(int maxN);
void HEAPfree(Heap h);
void HEAPfill(Heap h, Item val);
void HEAPsort(Heap h);
void HEAPdisplay(Heap h);
void HEAPify(Heap h, int i);
void HEAPbuild(Heap h);
```

ST.h

```
int GETindex(Key k);
ST STinit(int maxN);
void STfree(ST st);
int STcount(ST st);
void STinsert(ST st, Item val);
Item STsearch(ST st, Key k);
Item STselect(ST st, int r);
void STdelete(ST st, Key k);
void STdisplay(ST st);
```

BST.h

```
BST BSTinit();
void BSTfree(BST bst);
int BSTcount(BST bst);
int BSTempty(BST bst);
void BSTinsert_leafI(BST bst, Item x);
void BSTinsert_leafR(BST bst, Item x);
void BSTinsert_root(BST bst, Item x);
void BSTdelete(BST bst, Key k);
Item BSTsearch(BST bst, Key k);
Item BSTselect(BST bst, int r);
```

```
Item BSTmin(BST bst);
Item BSTmax(BST bst);
void BSTvisit(BST bst, int strategy);
Item BSTsucc(BST bst, Key k);
Item BSTpred(BST bst, Key k);
link rotL(link h);
link rotR(link h);
link partR(link h, int r);
link joinLR(link a, link b, link z);
void BSTbalance(BST bst);
```

IBST.h

```
void IBSTinit(IBST ibst);
void IBSTfree(IBST ibst);
int IBSTcount(IBST ibst);
int IBSTempty(IBST ibst);
void BSTinsert(IBST ibst, Item x);
link rotL(link h);
link rotR(link h);
link partR(link h, int r);
link joinLR(link a, link b, link z);
void IBSTdelete(IBST ibst, Item x);
Item IBSTsearch(IBST ibst, Item x);
void IBSTvisit(IBST ibst, int strategy);
```

Graph.h

```
Graph GRAPHinit(int V);
void GRAPHfree(Graph G);
void GRAPHload(FILE *fin);
void GRAPHstore(Graph G, FILE *fout);
void GRAPHedges(Graph G, Edge *a);
int GRAPHgetIndex(Graph G, char *label);
void GRAPHinsertE(Graph G, int id1, int id2, int wt);
void GRAPHremoveE(Graph G, int id1, int id2);
```

```
void GRAPHpath(Graph G, int id1, int id2);
void GRAPHpathH(Graph G, int id1, int id2);
void GRAPHsimpleDfs(Graph G, int id);
void GRAPHextendedDfs(Graph G, int id);
void GRAPHbfs(Graph G, int id);
void GRAPHdfs(Graph G, int id);
int GRAPHcc(Graph G);
int GRAPHscc(Graph G);
Graph GRAPHreverse(Graph G);
void GRAPHmstK(Graph G);
void GRAPHmstP(Graph G);
void GRAPHspD(Graph G, int id);
void GRAPHspBF(Graph G, int id);
void TSdfsR(Graph G, int v, int *ts, int *pre, int *time);
void DAGrts(Graph G);
```