Adatszerkezet

A hívás résztvevői: hívó, hívást fogadó, műhold, kapcsolat (2 műhold között).

A műholdak számát nem ismerjük fordítási időben, ezért a tárolásuk láncolt listával oldható meg hatékonyan. A műholdakat tartalmazó láncolt lista egy irányba láncolt és nem strázsás. Minden műhold listaelemről az abból a műholdból látható másikak adatai függnek egy újabb láncolt listán. Ezek a műhold kapcsolatai.

Ha az adatszerkezet felépült a fájlok beolvasásával, le kell rajta futtatni egy Dijkstra algoritmust. A Dijkstra futása kezdetekor (a hívó fölöttit kivéve) minden műhold eléretlen és befejezetlen. Az egyes műholdak hívótól való elérésének felső becslését nyilvántartjuk, és élmenti (vagyis kapcsolatmenti) javításokat végzünk. Minden műholdhoz nyilván kell tartani az a közvetlen szomszédját, ahonnan befejezetté válik, ez a beállító-ja (egyszerűbb a nyilvántartás, ha sikeres élmenti javításkor a “beállító” az a műhold lesz, amelyikből az élmenti javítást végeztük és külön adatban tároljuk a befejezettséget).

A Dijkstra futása közben egy lépésként egy műhold összes szomszédja felé végzünk élmenti javítást. Ehhez ismernünk kell egy adott műhold összes kapcsolatát, ez teszi indokolttá a kapcsolatok két oldali tárolását (vagyis az 1-es és 2-es műhold kapcsolatát nyilvántartjuk az 1-es és a 2-es műhold felől is).

Nyilván kell tartani a következőket egy adott műholdról: befejezett-e már, mennyi az eddig meghatározott legkisebb percdíjú összeköttetése a hívóval, és melyik műholddal közös közvetlen kapcsolata állította be (“Beállító”) a végleges (befejezése utáni) legrövidebb percdíjat a hívó és az adott műhold között. A beállító műholdat egy az arra a műhold típusra mutató pointer tartalmazza, ami nullpointer, ha még nincs elérve a műhold.

A KAPCS típus adatai:
a kapcsolódó műhold száma (int szam),
a kapcsolat percdíja (int dij),
a következő kapcsolatra mutató pointer (kapcs* kov).

A MUHOLD típus adatai:
a műhold száma (int szam),
a kapcsolatai láncolt listájának feje (kapcs* khead),
az elérése költségének felső becslése (int fb), (negatív, ha nincs elérve)
a következő műholdra mutató pointer (muhold* kov),
a beállító műholdra mutató pointer (muhold* beallito), (nullpointer, ha nincs elérve)
befejezettség (char bef, lehet: ’i’/’n’).

A műholdak láncolt listájának szerkezete:
láncolt lista feje: muhold* MHEAD
következő műhold pointere: nullpointer, ha ez az utolsó elem
rendezettség: a műholdak láncolt listájának rendezettsége lényegtelen, mivel futás közben a meghatározó tulajdonság a műholdak kapcsolata, ami nem feltétlenül sorba rendezhető.

Egy műholdon lógó kapcsolatok láncolt listájánal szerkezete:
láncolt lista feje: kapcs* khead (ez a muhold típus egyik paramétere)
következő kapcsolat pointere: nullpointer, ha nincs több kapcsolata az adott műholdnak

rendezettség: itt sem szükséges, mert a műholdak sorbarendezése a számuk alapján lenne lehetséges, a számuk pedig egy puszta elnevezés, ami a feladat megoldása szempontjából irreleváns.

Alap függvények:

Létrehoz egy kapcsolatok nélküli, eléretlen műholdakból álló, n-hosszú láncolt listát:
muhold* M_lista_Generator(int n)

Felszabadít egy kapcsolatok nélküli műholdlistát:
void M_free(muhold* MHEAD)

Felszabadítja az összes műhold összes kapcsolatát:
void K_free(muhold* MHEAD)

2 megadott műhold (n és k) között megadott percdíjú kapcsolatot hoz létre, vagy a már meglevő kapcsolat percdíját megváltoztatja, vagy kitörli a kapcsolatot, ha az új percdíj 0:
void osszekapcs(muhold* MHEAD, int n, int k, int ujdij)

Egy műhold (n) kapcsolatai mentén élmenti javítást végez:
void javito(muhold* MHEAD, int n)

Visszaadja az n számú műholdra mutató pointert:
muhold* M_kereso(muhold* MHEAD, int n)

Megmutatja, hogy vége van-e a keresésnek (int-et ad vissza, 0-t ha még nincs vége a keresésnek, 1-et ha megvan a legolcsóbb út a kezdő- és végpont között, 2-t ha minden műhold befejezett vagy eléretlen és a hívott fél műholdja eléretlen, vagyis nem lehet a hívó feleket összekötni):
int vege(muhold* MHEAD, int hivo, int hivott)

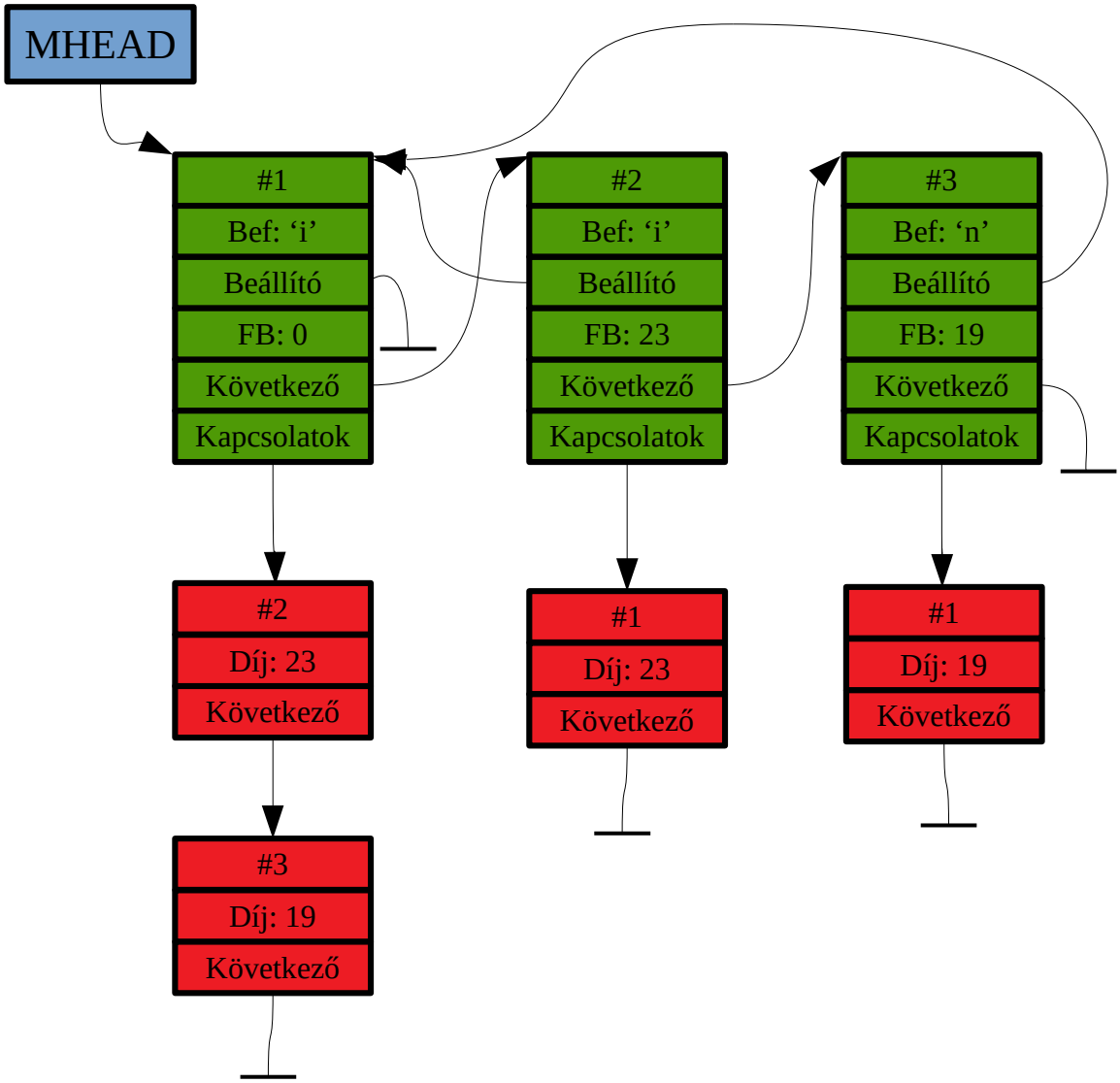
További függvények:

Felszabadítja a pointerrel mutatott műhold legelső kapcsolatát:
void K1_free(muhold* MHEAD, muhold* muh)

Felszabadítja az első műhold listaelemet egy nemüres, kapcsolatok nélküli műholdlistában:
void M1_free(muhold* MHEAD)

Élmenti javítást végez egy adott műhold (n) egy adott (pointerrel megadott) kapcsolata mentén:
void javito1(muhold* MHEAD, int n, kapcs* el)

Kiírja az adott elért műholdhoz vezető legolcsóbb úton szereplő műholdak számait és az út összköltségét:
void kiir(muhold* MHEAD, int n)


MHEAD
#1
Bef: ‘i’
Beállító
FB: 0
Következő
Kapcsolatok
#2
Bef: ‘i’
Beállító
FB: 23
Következő
Kapcsolatok
#3
Bef: ‘n’
Beállító
FB: 19
Következő
Kapcsolatok
#2
Díj: 23
Következő
#3
Díj: 19
Következő
#1
Díj: 23
Következő
#1
Díj: 19
Következő