

Karácsonyi bevásárlás

Felhasználói dokumentáció

Egy szöveges fájl egy bevásárlóközpont adatait tartalmazza: minden részlegről 4 sort az alábbi szerkezet szerint.

Részleg azonosítója (karakter (a-z))

A szomszédos részlegek betűjelei: egymás után, szóközzel elválasztva, 'X' jelöli a kijáratot (max. 26 féle karakter + 25 szóköz = max. 51 karakter hosszú)

A részlegen tartózkodó emberek száma (egész szám)

A részleg alapterülete négyzetméterben (valós érték)

Egy másik szöveges fájl a különféle ünnepi sütemények összetevőinek adatait tartalmazza: mindegyikről 2 sort az alábbi módon.

Összetevő azonosítója: a 2 valamely egész kitevőjű hatványa (egész szám)

Összetevő helye: a részleg azonosítója, ahol fellelhető (karakter (a-z))

Egy harmadik szöveges fájl recepteket tartalmaz, ennek minden sora az alábbi formájú.

Recept azonosítója (max. 20 karakter hosszú)/**Összetevők listája** (kettes számrendszerbeli, 16 jegyű szám) pl. ZSERBO/1000010111011111

Standard inputon érkezik egy recept azonosítója (pl. ISCHLER) és valamely részleg, mint kiindulási pont azonosítója (pl. d), szóközzel elválasztva.

Az két részleg távolságát nem a tényleges geometriai távolságok, hanem a bevásárlóközpont zsúfoltsága határozza meg (a vásárlók számának és a részleg alapterületének hányadosa)

Összetevők kódjai

1	liszt
2	cukor
4	tojás
8	tej
16	vaj
32	sütőpor
64	élesztő
128	vanília
256	csokoládé
512	mák
1024	lekvár
2048	marcipán
4096	méz
8192	fahéj
16384	szegfűszeg
32768	dió

A program a három fájl beolvasása és eltárolása után a standard inputon beérkező recept összetevőinek listáját és beszerzésüknek legrövidebb útját adja meg úgy, hogy a kezdőpont a beérkezett részleg, a végpont pedig mindenképp a kijárat. A standard outputon így egy bevásárlólista, illetve az érintendő részlegek betűjeleinek sorozata jelenik meg, szóközzel elválasztva. Ha az adott recept összetevői közül bármelyik hiányzik, akkor ezt a program közli.

Példa:

INPUT: *bevasarlokozpont.txt, receptek.txt, osszetevok.txt*

stdin: PISKOTA d

OUTPUT:

BEVASARLOLISTA

- liszt

- cukor

- tojas

Idealis utvonal: d a f b X

Példa bemeneti fájlokrabevasarlokozpont.txt

a
b c f
1625
32.5
b
a X d
150
10
d
b X
30
2.5
c
f a
660
20
f
c a
600
30

osszetevok.txt

2 f
1 b
2048 d
512 c
256 c
4 a
32 f
64 f

receptek.txt

PISKOTA/00000000000000111
ISCHLER/1010010100010011
MEZESKALACS/0111000010110111
ZSERBO/1000010111011111
MAKOSBEJGLI/0001011011011111
DIOSBEJGLI/1001000011011111
MAKOSGUBA/0000001011001011
DIOSGUBA/1000000011001011
LINZER/0000010000010111
KALACS/0000000001011111
MEZESKREMES/0001010110111111
MEZESPUSZEDLI/0111000000110111
VAJASKEKSZ/0000000110110111
VANILIASKIFLI/1000000010010011
MARCIPANOSKEKSZ/0000100010110111
LEKVAROSBUKTA/0000010001011111

A program nem ellenőrzi a bemeneti fájlok helyességét, ezek megfelelő létrehozása a felhasználó feladata.
A részlegeket tartalmazó fájl létrehozásánál különösen ügyeljünk arra, hogy a részlegek szomszédossága mindkét részlegnél megjelenjen!

Standard válaszadástól eltérő kimenetek

Hiba a(z) <fájlnév> fájl megnyitásakor

Az adott nevű fájl megnyitása sikertelen volt.

Ellenőrizze a fájl nevét, és hogy a futtatható állománnyal megegyező mappában van.

Hiba a(z) <fájlnév> fájl bezárasakor

Az adott nevű fájl bezárása sikertelen volt.

Sikertelen memoriafoglalás

A program nem tudott megfelelő dinamikus memóriát foglalni a működéshez, ekkor nem feltétlenül áll le, de a kimenet hibás lesz vagy végtelen ciklusba kerülhet, így mindenképp ajánlott a leállítása.

A megadott recept nem létezik

A program számára standard inputon beadott recept nem szerepel a receptek.txt fájlban felsorolt receptek között. Adjon meg egy létező receptet, vagy adja hozzá a keresni kívánt receptet a fájlhoz!

A megadott részleg nem létezik

A program számára standard inputon beadott kezdőpont nem szerepel a bevasarlokozpont.txt fájlban felsorolt részlegek között. Adjon meg egy létező részleget!

Programozói dokumentáció**Felépítendő adatszerkezetek**bevasarlokozpont.txt

4 sor → 1 részleg

struct reszleg

int id	'a' → 1, 'b' → 2 stb. 'X' → 0
int szomszedok[27]	'b d X f' → {1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0...}
double tomeg	emberek száma/alapterület

Tárolásuk kétstrázsás láncolt listában:

struct rszlelem

reszleg rszl	rszl.id szerint sorbarendezeve
struct rszlelem *next	következő elemre mutató pointer

receptek.txt

1 sor → 1 recept

struct recept

char nev[20+1]	pl. "ISCHLER"
int osszetevek[16]	1010010100010011 → {1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1}

Tárolásuk kétstrázsás láncolt listában:

struct rclelem

recept rec	
struct rclelem *next	következő elemre mutató pointer

osszetevek.txt

1 sor → 1 összetevő

int otidx	1→0, 2→1, 4→2, 8→3 stb. ($\log_2(\text{azonosító})$)
rszlelem *hely	Adott részlegre mutató pointer

Tárolásuk statikus tömbben: rszlelem* osszetevek[16] (pointerek tömbje)

osszetevek[otidx]: az adott részlegre mutató pointer

Algoritmus

Inicializáció

nev: karakter //részleg betűjele: a, b, c, ... X ha kijárat
 azon: egész //részleg azonosítója: 1, 2, 3, ... 0 ha kijárat
 idx: egész //részlegek listájában a részleg sorszáma, tömbjükben az indexe

Struktúrák

Részlegeket tároló struktúra létrehozása
 Részlegek listáját tároló struktúra létrehozása
 Recepteket tároló struktúra létrehozása
 Receptek listáját tároló struktúra létrehozása

Függvények

Segédfüggvények

Hibaüzenet (error_message)

BE: fname (sztring) a hibát adó fájl neve, mode (karakter) a fájlkezelés típusa o - megnyitás, c - bezárás, m - memóriakezelés
 MŰKÖDÉS: Kiírja a kapott névvel és móddal a hibaüzenetet stderr-ra

Névből azonosító (nametoid)

BE: name (nev) részleg betűjele
 KI: azon beadott nevű részleg azonosítója

Koordinátákból index (coord)

BE: i (egész) oszlop indexe, j (egész) sor indexe, meret (egész) sorok és oszlopok száma
 KI: egész egydimenziós „mátrixtömb” indexe

Szükségesek sorszámból permutálandók idx-tömbje (kellidx_to_permin)

BE: kell (idx tömb) érintendő részlegek, lk (egész) tömb hossza, start_idx (idx) kiindulási részleg sorszáma, perm (idx tömb címe) érintendő részlegek közül a permutálandók
 MŰKÖDÉS: kell bejárása → perm mérete, perm helyfoglalása (sikertelen: [Hibaüzenet](#)), kell bejárásával perm feltöltése
 KI: egész permutálható tömb hossza, paraméterlistán kapott perm feltöltve

Tömb permutációja rekurzívan, fájlba írva (permutal)

BE: t (egész tömb) permutálandó tömb, start (egész) kezdőpont indexe, stop (egész) végpont indexe, out (fájl) kiírás helye
 MŰKÖDÉS: t összes permutációjának rekurzív kialakítása és kiírása f-be

Útvonal hosszának kiszámítása (uthossz)

BE: be (idx tömb) útvonal részlegeinek tömbje, l (egész) tömb mérete, start_idx (idx) kiindulási részleg sorszáma, distm (valós tömb) távolságokat tartalmazó egydimenziós „mátrixtömb”, n (egész) mátrix mérete
 MŰKÖDÉS: hossz= start-útvonal távolság + útvonal hossz + útvonal-kijárat távolság
 KI: valós útvonal hossza

Minimális hosszúságú út keresése fájlból (minutkeres)

BE: utak (fájl) útvonalakat tartalmazó (bináris) fájl, l (egész) egy útvonal elemszáma, start_idx (idx) kiindulási részleg sorszáma, distm (valós tömb) távolságokat tartalmazó egydimenziós „mátrixtömb”, n (egész) mátrix mérete
 MŰKÖDÉS: AMÍG utakból olvasás sikeres, minimumkeresés ([Útvonal hosszának kiszámítása](#))
 KI: idx tömb minimális hosszúságú útvonal részlegek sorszámaiból

Részlegeken dolgozó függvények

Üres részleglista létrehozása (rszl_ures)

MŰKÖDÉS: létrehozza (sikertelen: [Hibaüzenet](#)) az első és a hátsó strázsát, illetve a kijáratot jelképező részleget, azonosítóikat a rendezéshez ideális értékekkel inicializálja és összeláncol

KI: **részleg listaelem** lista feje

Részleg létrehozása és rendezve beszúrása listába (rszl_ujbe)

BE: **ID** (nev) részleg betűjele, **szomszedok** (51 karakteres sztring) szomszédos részlegek felsorolása, **emberek** (egész) részlegen lévő emberek száma, **terulet**(valós) részleg alapterülete, **fej** (részleg listaelem) részlegek listája

MŰKÖDÉS: megadott adatokból létrehoz (sikertelen: [Hibaüzenet](#)) egy részleg listaelemet és azt azonosító szerint rendezve, a listát bejárva beszúrja a listába

Részleg címe azonosítóból (rszl_nevtoptr)

BE: **betu** (nev) részleg betűjele, **fej** (részleg listaelem) részlegek listája

MŰKÖDÉS: [Névből azonosító](#)(betu), lista végigjárásával kikeresi az adott azonosítójú részleget

KI: **részleg listaelem** megadott nevű részleg

Két szomszédos részleg távolsága (rszl_dist)

BE: **r1** (idx) egyik részleg, **r2** (idx) másik részleg, **fej** (részleg listaelem) részlegek listája

MŰKÖDÉS: a két részleget a listából kikeresve ellenőrzi szomszédosságukat

KI: **valós** két részleg tömegadatainak összege, ∞ ha nem szomszédosak

Részlegek listájából tömb inicializálása (rszl_maketomb)

BE: **n** (egész) a létrehozandó tömb mérete (=részlegek száma)

MŰKÖDÉS: megadott méretű tömböt foglal (sikertelen: [Hibaüzenet](#)), a kijárat helyét 1-es értékkel, a tömb további részeit 0-ra inicializálja

KI: **egész tömb** címe

Részleg betűjele idx-ből (rszl_idxtoname)

BE: **r_idx** (idx) részleg sorszáma, **fej** (részleg listaelem) részlegek listája

MŰKÖDÉS: a listát végigjárva kikeresi a megfelelő sorszámú elemet, annak azonosítójából kiszámolja a nevét

KI: **nev** a részleg betűjele

Részleglista felszabadítása (rszl_freelist)

BE: **fej** (részleg listaelem) részlegek listája

MŰKÖDÉS: a listát két pointerrel végigjárva felszabadítja azt

Recepteken dolgozó függvények

Üres receptlista létrehozása (rc_ures)

MŰKÖDÉS: létrehozza (sikertelen: [Hibaüzenet](#)) az első és a hátsó strázsát és összeláncolja őket

KI: **recept listaelem** lista feje

Recept létrehozása és beszúrása lista elejére (rc_ujbe)

BE: **nev** (20 karakteres sztring) recept neve, **osszetevek** (16 karakteres sztring) 1/0 sorozat, **fej** (recept listaelem) receptek listája

MŰKÖDÉS: megadott adatokból létrehoz (sikertelen: [Hibaüzenet](#)) egy recept listaelemet és beszúrja a lista elejére

Recept címe névből (rc_nametoptr)

BE: **fej** (recept listaelem) receptek listája, **rcp** (20 karakteres sztring) recept neve

MŰKÖDÉS: listát bejárva kikeresi a megadott nevű receptet

KI: **recept listaelem** adott nevű recept

Receptlista felszabadítása (rc_freelist)

BE: fej (recept listaelem) részlegek listája

MŰKÖDÉS: a listát két pointerrel végigjárva felszabadítja azt

Összetevőkön dolgozó függvények

Nem használt összetevők törlése a tömbökből (ot_removeunused)

BE: **kell_rc** (recept listaelem) bevásárlandó recept, **osszetevek** (recept listaelem tömb címe) összetevők helyei, **ot_nevek** (sztring tömb címe) összetevők nevei

MŰKÖDÉS: **kell_rc** összetevőit bejárva kinullázza a recepthez nem szükséges helyeken lévő bejegyzéseket

KI: **egész 0** ha sikeres, **1** ha az adott recept összetevői nem találhatók meg az adott bevásárlóközpontban

Beolvasófüggvények

bevasarlokozpont.txt egy adatsorának beolvasása (read_bevkozpont)

BE: **in** (fájl) bemeneti fájl, **ID** (nev címe) részleg neve, **szomsz** (sztring címe) részleg szomszédjai, **fo** (egész címe) részlegen tartózkodók száma, **t** (valós címe) részleg alapterülete

MŰKÖDÉS: a megadott fájlból a megadott címekre olvas be adatot a fájlformátumnak megfelelően

KI: **egész 1** ha sikeres, **-1** ha nem

receptek.txt egy adatsorának beolvasása (read_rctek)

BE: **in** (fájl) bemeneti fájl, **nev** (sztring címe) recept neve, **otk** (sztring címe) összetevők 1/0

MŰKÖDÉS: a megadott fájlból a megadott címekre olvas be adatot a fájlformátumnak megfelelően

KI: **egész 1** ha sikeres, **-1** ha nem

osszetevek.txt egy adatsorának beolvasása (read_otk)

BE: **in** (fájl) bemeneti fájl, **otidx** (egész címe) összetevő azonosítója, **otreszleg** (nev címe) összetevő helye

MŰKÖDÉS: a megadott fájlból a megadott címekre olvas be adatot a fájlformátumnak megfelelően

KI: **egész 1** ha sikeres, **-1** ha nem

stdin adatok beolvasása (read_stdin)

BE: **recept** (sztring címe) bevásárlandó recept, **start** (nev címe) kiindulási részleg

MŰKÖDÉS: beolvassa a megadott címekre a standard inputra érkező adatokat

A program logikai elemei

bevkozpont.txt feldolgozása (process_bevkozpont)

BE: **fajlnev** (sztring) bemeneti fájl neve, **num_reszlegek** (egész címe) részlegek száma, **fej** (részleg listaelem) részlegek listája

MŰKÖDÉS: Fájl megnyitása, sikeresség ellenőrzése

[Üres részleglista létrehozása](#)

AMÍG [bevasarlokozpont.txt egy adatsorának beolvasása](#) sikeres,

[Részleg létrehozása és rendezve beszúrása listába](#), **num_reszlegek** növelése

Fájl bezárása, sikeresség ellenőrzése

KI: **egész 0**, sikertelen fájlkezelés esetén **1** és [Hibaüzenet](#)

receptek.txt feldolgozása (process_receptek)

BE: **fajlnev** (sztring) bemeneti fájl neve, **fej** (recept listaelem) receptek listája

MŰKÖDÉS: Fájl megnyitása, sikeresség ellenőrzése

[Üres receptlista létrehozása](#)

AMÍG [receptek.txt egy adatsorának beolvasása](#) sikeres,

Recept létrehozása és beszúrása lista elejére

Fájl bezárása, sikeresség ellenőrzése

KI: egész 0, sikertelen fájlkezelés esetén 1 és [Hibaüzenet](#)

osszetevek.txt feldolgozása (process_osszetevek)

BE: fajlnev (sztring) bemeneti fájl neve, bevkp (részleg listaelem) részlegek listája, osszetevek (részleg listaelem tömb)

MŰKÖDÉS: Fájl megnyitása, sikeresség ellenőrzése

osszetevek feltöltése NULL-pointerekkel

AMÍG [osszetevek.txt egy adatsorának beolvasása](#) sikeres,

[Részleg címe azonosítóból](#) eltávolítása a megfelelő indexen

Fájl bezárása, sikeresség ellenőrzése

KI: egész 0, sikertelen fájlkezelés esetén 1 és [Hibaüzenet](#)

stdin feldolgozása (process_stdin)

BE: bevkp (részleg listaelem) részlegek listája, receptek (recept listaelem) receptek listája, start (azon címe) kiindulási részleg azonosítója, recept (sztring címe) bevásárlandó recept

MŰKÖDÉS: [stdin adatok beolvasása](#), helyességük ellenőrzése listák bejárásával és kereséssel

KI: egész 0, ha nem értelmezhető valamelyik adat 1

Szomszédsági mátrix létrehozása (make_adjmatrix)

BE: n (egész) mátrix mérete, fej (részleg listaelem) részlegek listája

MŰKÖDÉS: helyet foglal a mátrixnak (sikertelen: [Hibaüzenet](#))

Sorokat és oszlopokat bejárva feltölti a mátrixot

$m[\text{Koordinátákból index}] \leftarrow \text{Két szomszédos részleg távolsága}$

KI: valós tömb mátrix

Floyd-algoritmus (floyd)

BE: dist_m (valós tömb) szomszédsági mátrix, n (egész) mátrix mérete

MŰKÖDÉS: Floyd-algoritmus működése szerint

Utak fájljának megírása (write_utak)

BE: kell_rszl (egész tömb) részlegek 1/0 tömbje, n (egész) tömb mérete, start_idx (idx) start részleg sorszáma, l (egész címe) kiírt adatsor hossza

MŰKÖDÉS: Fájl megnyitása, sikeresség ellenőrzése

$l \leftarrow \text{Szükségesek sorszámból permutálandók idx-tömbje}$

[Tömb permutációja rekurzívan, fájlba írva](#)

Fájl bezárása, sikeresség ellenőrzése

KI: egész 0, sikertelen fájlkezelés esetén 1 és [Hibaüzenet](#)

Utak fájljának feldolgozása (process_utak)

BE: l (egész) adatsor hossza, start_idx (idx) start részleg sorszáma, distm (valós tömb) távolságok mátrixa, n (egész) mátrix mérete, minut (idx tömb címe) minimális hosszúságú út

MŰKÖDÉS: Fájl megnyitása, sikeresség ellenőrzése

$\text{minut} \leftarrow \text{Minimális hosszúságú út keresése fájlból}$

Fájl bezárása, sikeresség ellenőrzése

KI: egész 0, sikertelen fájlkezelés esetén 1 és [Hibaüzenet](#)

Memória felszabadítása (free_memory)

BE: rszlfej (részleg listaelem) részlegek listája, rcfej (recept listaelem) receptek listája, kell_rszl (egész tömb címe) részlegek tömbje, dist (valós tömb címe) mátrix, minut (idx tömb címe) minimális hosszúságú út

MŰKÖDÉS: amennyiben léteznek, [Részleglista felszabadítása](#), [Receptlista felszabadítása kapott](#) címeken lévő memória felszabadítása

Konstansok és változók

```

bevkp_nev (sztring)           //részlegek adatait tartalmazó fájl neve
rec_nev (sztring)             //recepteket tartalmazó fájl neve
otk_nev (sztring)             //összetevők adatait tartalmazó fájl neve
ot_nevek (sztringek tömbje)   //összetevők neveinek tárolására

num_reszlegek (egész)         //részlegek száma
bevkp (részleg listaelem)     //részlegek listája DINAMIKUS
receptek (recept listaelem)   //receptek listája DINAMIKUS
osszetevok[16] (részleg listaelem tömb) //összetevők helyét tartalmazó tömb
start (azon)                  //kiindulási részleg azonosítója
beRCP (sztring)               //megvásárlandó recept beolvasott neve
dist (valós tömb)             //részlegek szomszédsági mátrixát tároló tömb (egydimenziós,
                                kétdimenziós indexeléshez külön függvény) DINAMIKUS

kell_rc (recept listaelem)     //bevásárlandó recept címe
kell_rszl (egész tömb)        //részlegek tömbje 1-szükséges, 0-nem szükséges DINAMIKUS
start_idx (idx)               //a start részleg helye a tömbben
minut(idx tömb)               //minimális hosszúságú út részlegek sorszámaként tárolva
                                DINAMIKUS

l_kell (egész)                //permutált tömb mérete

```

Adatok feldolgozása

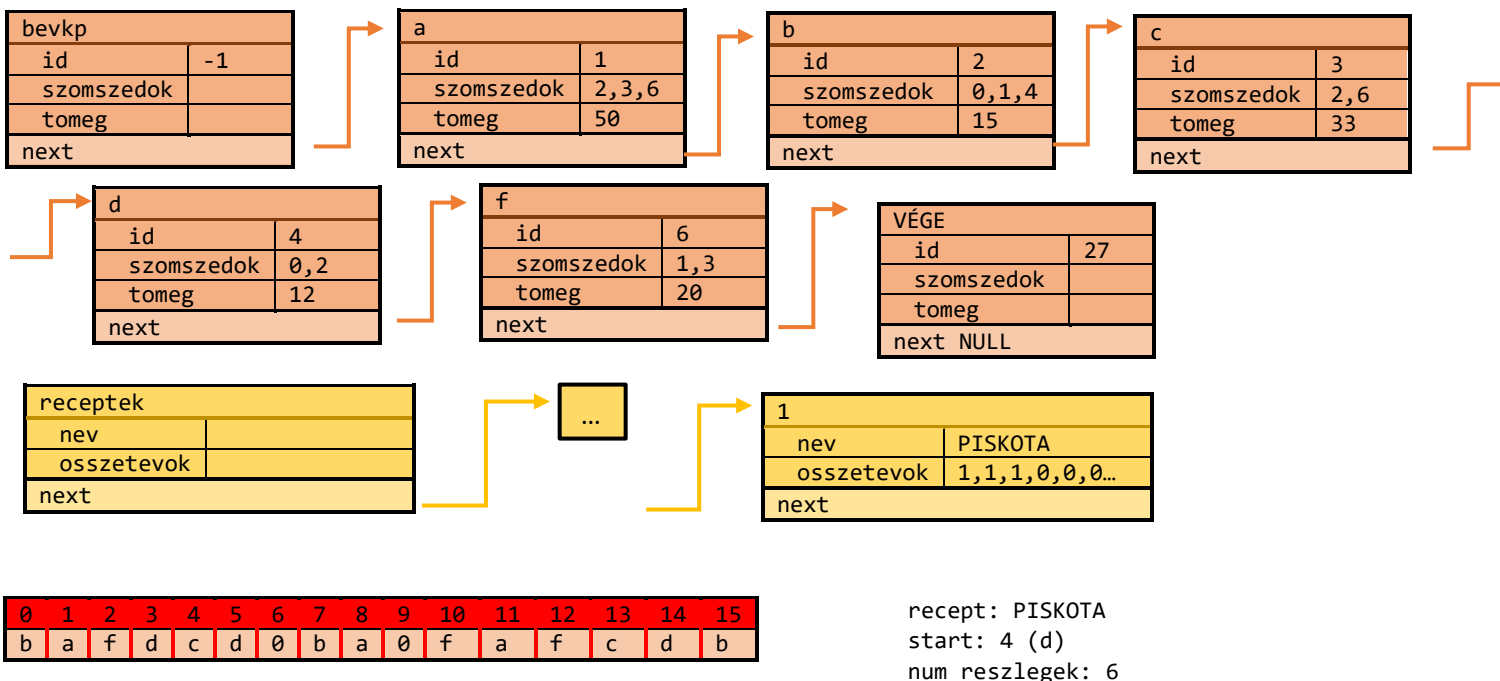
[bevkozpont.txt feldolgozása](#)

[receptek.txt feldolgozása](#)

[osszetevok.txt feldolgozása](#)

[stdin feldolgozása](#)

Sikertelen fájlkezelés esetén [Memória felszabadítása](#) és program befejezése



Eredmény kiszámítása

Gráf létrehozása és legrövidebb utak

dist ← [Szomszédsági mátrix létrehozása](#)

[Floyd-algoritmus](#)

Így megkapjuk bármely két részleg között a legrövidebb út hosszát. Az út csúcsainak eltárolása igen nehéz lenne, ez maradjon a program egyik továbbfejlesztési lehetősége. Így a kimeneten ugyan az érintendő részlegek fognak megjelenni, de csak azok, ahol vennünk is kell valamit.

//dist így bármely két részleg közt a legrövidebb út hosszát tartalmazza

dist	0 (X)	1 (a)	2 (b)	3 (c)	4 (d)	5 (f)
0 (X)	0	∞	15	∞	12	∞
1 (a)	∞	0	65	83	∞	70
2 (b)	15	65	0	∞	27	∞
3 (c)	∞	83	∞	0	∞	53
4 (d)	12	∞	27	∞	0	∞
5 (f)	∞	70	∞	53	∞	0

Floyd →

dist	0 (X)	1 (a)	2 (b)	3 (c)	4 (d)	5 (f)
0 (X)	0	80	15	163	12	150
1 (a)	80	0	65	83	92	70
2 (b)	15	65	0	148	27	135
3 (c)	163	83	148	0	175	53
4 (d)	12	92	27	175	0	162
5 (f)	150	70	135	53	162	0

Szükséges részlegek

kell_rszl ← [Részlegek listájából tömb inicializálása](#)

kell_rc ← [Recept címe névből](#)

[Nem használt összetevők törlése a tömbökből](#)

Összetevő hiánya esetén KI hibaüzenet, [Memória felszabadítása](#)

	0 (X)	1 (a)	2 (b)	3 (c)	4 (d)	5 (f)
kell_r	1	1	1	0	1	1

curr ← **bevkp**->next->next és **j**(egész) =1-től,

AMÍG **curr**->next nem NULL,

curr-t a köv. elemre léptetve és **j**-t egyesével növelve *//részlegek végigjárása*

i(egész) = 0-tól, AMÍG **i**<16, egyesével

HA **curr** = **osszetevok**[**i**]

kell_r[**j**] ← 1

HA **curr**->rszl.id = **start**

kell_r[**j**] ← 1

start_idx ← **j**

//részleg címe szükséges összetevők címei közt

//az adott részleg szükséges

//kiindulási pont, mindenképp kell

Legrövidebb út

Egy olyan súlyozott teljes gráfot építünk fel, melynek csúcsai az elérendő részlegek, éleinek hossza pedig a két részleg közötti legrövidebb út hossza. Az előző lépésben felépített gráfon a legrövidebb olyan utat kell megtalálni, ami **start**-ból indul és **X**-ben ér véget és érinti az összes csúcsot, de mindegyiket csak egyszer. Ez a probléma valójában az utazó ügynök problémájának (TSP - Travelling Salesman Problem) egyik alelete, ahol nem Hamilton-kört, hanem csak -utat keresünk. Erről [bizonyított](#), hogy nem egyszerűsíti a problémát. TSP megoldására kevés csúcs esetén alkalmazható kombinatorikus módszer, avagy egyszerűen megvizsgáljuk a gráf összes Hamilton-útját, és meghatározzuk, melyik a legrövidebb. Jelen esetben egy receptben maximum 16 összetevő lehet, azonban a bemeneti fájl receptjei közt a legbonyolultabbak is legfeljebb 10 összetevőből állnak, ez azt jelenti, hogy a kiindulási ponton kívül legfeljebb 10 részleget kell érintenünk. Ez legfeljebb $10! = 3\,628\,800$ út közti minimumkeresést jelent, ami egy számítógép számára még aránylag rövid időn belül teljesíthető. Ekkora területet azonban nem szeretnék dinamikusan foglalni, nagyobb bementek esetén elképzelhető, hogy nem is áll rendelkezésre ennyi dinamikus memória, ezért a lehetséges utakat inkább kiírom egy bináris fájlba, majd azokat visszaolvasva megkeresem a legrövidebbet, ez lesz a program kimenete. (csökkenthető lenne a fájl méret, ha nem egészekként (4 bájt), hanem karakterekként (1 bájt) tárolnám a sorszámokat, hiszen azok maximumértéke 26, ez bőven az ábrázolási tartományon belül van)

//dist jelenleg a teljes bevásárlóközponttra tartalmazza a távolságadatokat, kell_r pedig azokon az indexeken tartalmaz 1-est, amely indexű részlegek elérendők. Nem szükséges egy másik mátrix felépítése kizárólag az elérendő részlegek számára, egyszerűen nem használjuk a többi bejegyzést

Utak fájljának megírása

Utak fájljának feldolgozása

Sikertelen fájlkezelés esetén [Memória felszabadítása](#) és program befejezése

Fájlban tárolt útvonal	Jelentése	Útvonal hossza	Teljes út hossza	minut[] =
1 2 5	a b f	$65+135= 200$	$92 +200+150= 442$	= {4, 1, 5, 2, 0}
1 5 2	a f b	$70+135= 205$	$92 +205+15 = 312$	
2 1 5	b a f	$65+70 = 135$	$27 +135+150= 312$	//d a f b X
2 5 1	b f a	$135+70= 205$	$27 +205+80 = 312$	
5 1 2	f a b	$70+65 = 135$	$162+135+15 = 312$	
5 2 1	f b a	$135+65= 200$	$162+200+80 = 442$	

Eredmény megjelenítése

Kiír: BEVASARLOLISTA

i(egész) = 0-tól, AMÍG i<16, egyesével

HA **osszetevok[i]** nem NULL

Kiír: - **osszetevok[i]**

Kiír: Ideális útvonal:

i(egész) = 0-tól, AMÍG i<1_kell+2, egyesével

Kiír: [Részleg betűjele idx-ből](#)

BEVASARLOLISTA

-liszt

-cukor

-tojas

Ideális utvonal: d a f b X

Memória felszabadítása

[Memória felszabadítása](#) és program befejezése

Teszt dokumentáció

Debugging

Debugolás során a felhasználói dokumentációban példaként mutatott bementi fájlokat használtam, ezek feldolgozása még manuálisan is könnyen kivitelezhető, a programozói dokumentációban szerepel.

Használt fájlok: 0_bevasarlokozpont.txt, 0_osszetevek.txt, receptek.txt

Fájlkezelés, memóriafoglalás, hibaüzenetek

Beolvasás

Nemlétező fájlnevet adva a programnak a hibaüzenet megjelenésének ellenőrzése

```
c:\Prog\nagyhazi\karacsony_bev.exe
Hiba a(z) rcptek.txt fajl megnyitasakor_
```

```
c:\Prog\nagyhazi\karacsony_bev.exe
Hiba a(z) 3_bevasarlokozpont.txt fajl megnyitasakor_
```

```
c:\Prog\nagyhazi\karacsony_bev.exe
Hiba a(z) osszetevek.bin fajl megnyitasakor_
```

```
c:\Prog\nagyhazi\karacsony_be...
PISKOTA d
Hiba a(z) perm.bin fajl megnyitasakor_
```

Bezárás

Nem tudtam olyan helyzetet előállítani, amiben a fájl bezárása sikertelen lenne, de tekintve, hogy a hibaüzenet kiírását ugyanaz a függvény hajtja végre, a program ezen részét helyesnek tekintem.

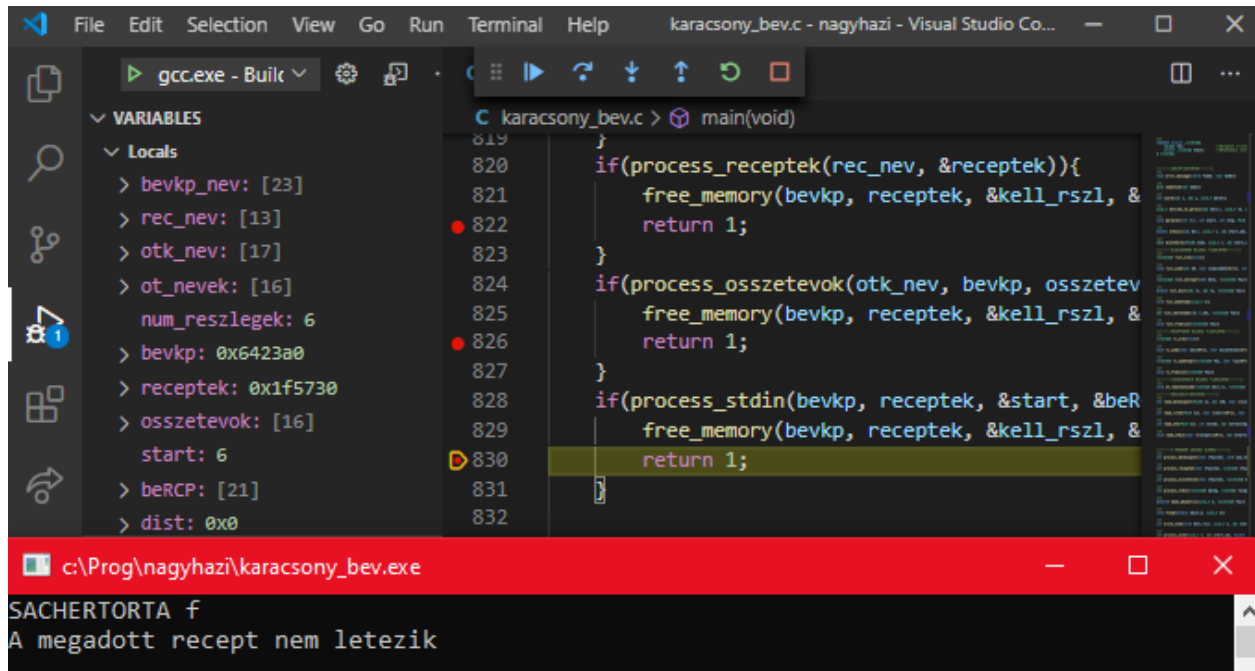
Memóriafoglalás

Csak mesterségesen, a változó értékének manuális átírásával tudtam olyan helyzetet előállítani, ahol a memóriafoglalás „sikertelen”, ekkor azonban működött a hibaüzenet. A program személyi számítógépen történő futtatása esetén ilyen helyzet egyébként sem is valószínű.

```
File Edit Selection View Go Run Terminal Help karacsony_bev.c - nagyhazi - Visual Studio Co...
gcc.exe - Built
VARIABLES
  Locals
    > fej: 0x0
    > veg: 0x642610
    > exit: 0x642880
karacsony_bev.c > rszl_ures(void)
104 > /...
172 > idx* minuterkes(FILE* utak, size_t l, idx start_id
200
201 //-----RÉSZLEGEKEN DOLGOZÓ FÜGGVÉNYEK-----//
202 > /** ...
207 rszlelem* rszl_ures(void){
208   rszlelem *fej= (rszlelem*)malloc(sizeof(rszlel
209   rszlelem *veg= (rszlelem*)malloc(sizeof(rszlel
210   rszlelem *exit= (rszlelem*)malloc(sizeof(rszle
211   if(fej==NULL||veg==NULL||exit==NULL) //sikert
212     error_message("-", 'm');
213   return NULL;
214
215   fej->rszl.id=-1; //rendezést segíti, 0
c:\Prog\nagyhazi\karacsony_bev.exe
Hiba memoriafoglalas soran_
```

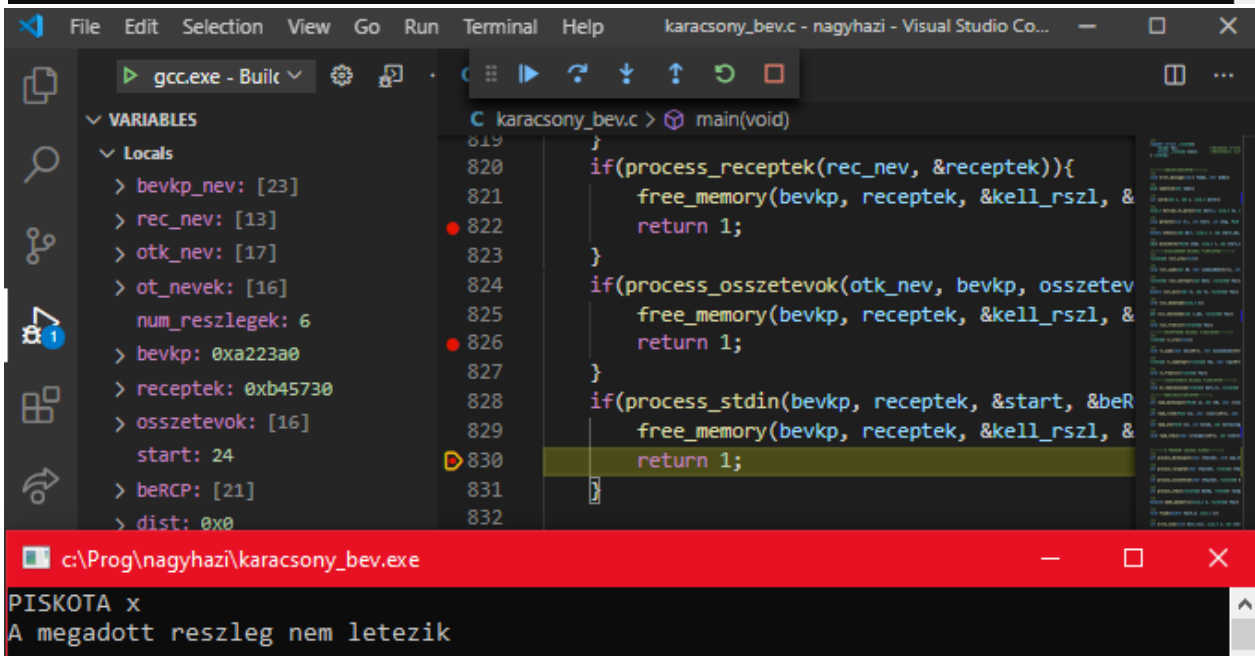
Felhasználói input

A felhasználó által megadott információ három esetben lehet helytelen, ha a megadott recept vagy a megadott részleg nem létezik, illetve ha nem áll rendelkezésre az összes összetevő. Mindhárom esetet a program közli a felhasználóval, majd leáll.



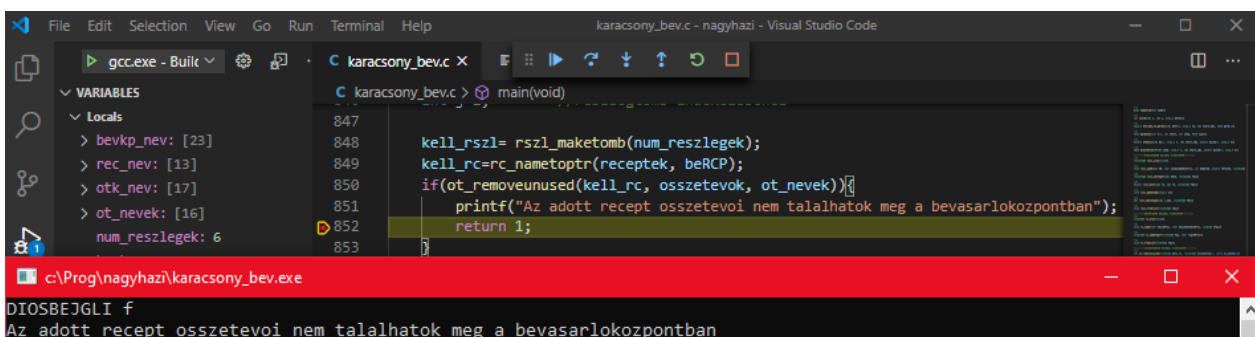
The screenshot shows the Visual Studio Code interface with the file `karacsony_bev.c` open. The `main` function is visible, and the program has crashed. The error message in the terminal is:

```
SACHERTORTA f
A megadott recept nem letezik
```



The screenshot shows the Visual Studio Code interface with the file `karacsony_bev.c` open. The `main` function is visible, and the program has crashed. The error message in the terminal is:

```
PISKOTA x
A megadott reszleg nem letezik
```



The screenshot shows the Visual Studio Code interface with the file `karacsony_bev.c` open. The `main` function is visible, and the program has crashed. The error message in the terminal is:

```
DIOSBEJGLI f
Az adott recept osszetevoi nem talalhatok meg a bevasarlokozpontban
```

Adatszerkezetek

```

bevkp: 0x9123a0
  rszl
    id: -1
    szomszedok
      tomeg: 5.2285141982483265e+54
    next: 0x912880
  rszl
    id: 0
    szomszedok
      tomeg: 0
    next: 0x913b20
  rszl
    id: 1
    szomszedok
      tomeg: 50
    next: 0x913d90
  rszl
    id: 2
    szomszedok
      tomeg: 15
    next: 0xa35250
  rszl
    id: 3
    szomszedok
      tomeg: 33
    next: 0xa34fe0
  rszl
    id: 4
    szomszedok
      tomeg: 12
    next: 0xa354c0
  rszl
    id: 6
    szomszedok
      tomeg: 20
    next: 0x912610
  rszl
    id: 27
    szomszedok
      tomeg: 5.2285141982483265e+54
    next: 0x0

receptek: 0xa35730
  rec
    next: 0xa37e00
  rec
    next: 0xa37c30
  nev
    [0]: 77 'H'
    [1]: 65 'A'
    [2]: 82 'R'
    [3]: 67 'C'
    [4]: 73 'I'
    [5]: 80 'P'
    [6]: 65 'A'
    [7]: 78 'N'
    [8]: 79 'O'
    [9]: 83 'S'
    [10]: 75 'K'
    [11]: 69 'E'
    [12]: 75 'K'
    [13]: 83 'S'
    [14]: 90 'Z'
    [15]: 0 '\000'
    [16]: 0 '\000'
    [17]: 0 '\000'
    [18]: 0 '\000'
    [19]: 0 '\000'
    [20]: 0 '\000'
  osszetevek
    [0]: 1
    [1]: 1
    [2]: 1
    [3]: 0
    [4]: 1
    [5]: 1
    [6]: 0
    [7]: 1
    [8]: 0
    [9]: 0
    [10]: 0
    [11]: 1
    [12]: 0
    [13]: 0
    [14]: 0
    [15]: 0
  next: 0xa379e0
  rec
  nev
  osszetevek

osszetevek: [16]
  [0]: 0x913d90
  [1]: 0xa354c0
  [2]: 0x913b20
  rszl
    id: 1
    szomszedok
      tomeg: 50
    next: 0x913d90
  [3]: 0x0
  [4]: 0x0
  [5]: 0xa354c0
  rszl
    id: 6
    szomszedok
      tomeg: 20
    next: 0x912610
  [6]: 0xa354c0
  [7]: 0x0
  [8]: 0xa35250
  rszl
    id: 3
    szomszedok
      tomeg: 33
    next: 0xa34fe0
  [9]: 0xa35250
  [10]: 0x0
  [11]: 0xa34fe0
  rszl
    id: 4
    szomszedok
      tomeg: 12
    next: 0xa354c0
  [12]: 0x0
  [13]: 0x0
  [14]: 0x0
  [15]: 0x0

```

Az adatszerkezetek létrejöttét debugging mode-ban nyomon lehet követni, itt ellenőrizhető, hogy a megfelelő adatszerkezetek jöttek létre.

A program fontosabb változói

A main-ben definiált változók ellenőrzésével megbizonyosodhatunk az algoritmus helyes működéséről.

num_reszlegek

X a b c d f – 6 részleg

start, start_idx

d – 4 ill. 4.

beRCP, kell_rc

PISKOTA

kell_rszl

a, b, f összetevőkhöz,

d start, X kijárat

num_reszlegek: 6

start_idx: 4

kell_rszl[0]: 1
kell_rszl[1]: 1
kell_rszl[2]: 1
kell_rszl[3]: 0
kell_rszl[4]: 1
kell_rszl[5]: 1

start: 4

beRCP: [21]

[0]: 80 'P'
[1]: 73 'I'
[2]: 83 'S'
[3]: 75 'K'
[4]: 79 'O'
[5]: 84 'T'
[6]: 65 'A'
[7]: 0 '\000'

c:\Prog\nagyhazi\karacsony_bev.exe

PISKOTA d

kell_rc: 0x1f5bd0

rec

nev

[0]: 80 'P'
[1]: 73 'I'
[2]: 83 'S'
[3]: 75 'K'
[4]: 79 'O'
[5]: 84 'T'
[6]: 65 'A'
[7]: 0 '\000'
[8]: 0 '\000'

c:\Prog\nagyhazi\karacsony_bev.exe

PISKOTA d

l_kell

a, b, f – 3

l_kell: 3

minut

d, a, f, b, X – 4, 1, 5, 2, 0

dist

```
minut[0]: 4
minut[1]: 1
minut[2]: 5
minut[3]: 2
minut[4]: 0
```

ot_nevek

liszt,

cukor,

tojás

```
ot_nevek: [16]
> [0]: 0x4086e1 "liszt"
> [1]: 0x4086e7 "cukor"
> [2]: 0x4086ed "tojás"
> [3]: 0x0
> [4]: 0x0
> [5]: 0x0
> [6]: 0x0
> [7]: 0x0
> [8]: 0x0
```

dist	0 (X)	1 (a)	2 (b)	3 (c)	4 (d)	5 (f)
0 (X)	0	∞	15	∞	12	∞
1 (a)	∞	0	65	83	∞	70
2 (b)	15	65	0	∞	27	∞
3 (c)	∞	83	∞	0	∞	53
4 (d)	12	∞	27	∞	0	∞
5 (f)	∞	70	∞	53	∞	0

dist	0 (X)	1 (a)	2 (b)	3 (c)	4 (d)	5 (f)
0 (X)	0	80	15	163	12	150
1 (a)	80	0	65	83	92	70
2 (b)	15	65	0	148	27	135
3 (c)	163	83	148	0	175	53
4 (d)	12	92	27	175	0	162
5 (f)	150	70	135	53	162	0

```
dist[0]: 0
dist[1]: inf dist[6+1]: 0
dist[2]: 15 dist[6+2]: 65 dist[12+2]: 0
dist[3]: inf dist[6+3]: 83 dist[12+3]: inf dist[18+3]: 0
dist[4]: 12 dist[6+4]: inf dist[12+4]: 27 dist[18+4]: inf dist[24+4]: 0
dist[5]: inf dist[6+5]: 70 dist[12+5]: inf dist[18+5]: 53 dist[24+5]: inf dist[30+5]: 0
```

```
dist[0]: 0
dist[1]: 80 dist[6+1]: 0
dist[2]: 15 dist[6+2]: 65 dist[12+2]: 0
dist[3]: 163 dist[6+3]: 83 dist[12+3]: 148 dist[18+3]: 0
dist[4]: 12 dist[6+4]: 92 dist[12+4]: 27 dist[18+4]: 175 dist[24+4]: 0
dist[5]: 150 dist[6+5]: 70 dist[12+5]: 135 dist[18+5]: 53 dist[24+5]: 162 dist[30+5]: 0
```

Kimenet, eredmény megjelenítése

Végül meg is jelenik a megfelelő kimenet, ezzel megbizonyosodtam, hogy a program valóban működik.

Memóriaszivárgás

debugmalloc.h segédkönyvtárral

```
c:\Prog\nagyhazi\karacsony_bev.exe
Adjon meg egy receptet es egy kiindulasi pontot!
<RECEPT> <start>: PISKOTA d

BEVASARLOLISTA
- liszt
- cukor
- tojas
-----
Idealis utvonal: d a f b X
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
3: cppdbg: karacsony_b

PS C:\Prog\nagyhazi> & 'c:\Users\Andi\.vscode\extensions\ms-vscode.cpptools-1.1.3\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-rlhcr2hs.0kw' '--stdout=Microsoft-MIEngine-Out-vkccyytz.0tu' '--stderr=Microsoft-MIEngine-Error-gthkmblo.50l' '--pid=Microsoft-MIEngine-Pid-hucc2rbv.kvc' '--dbgExe=C:\Program Files\mingw-w64\x86_64-8.1.0-posix-seh-rt_v6-rev0\mingw64\bin\gdb.exe' '--interpreter=mi'
Adjon meg egy receptet es egy kiindulasi pontot!
<RECEPT> <start>: PISKOTA d

BEVASARLOLISTA
- liszt
- cukor
- tojas
-----
Idealis utvonal: d a f b X
*****
* Debugmalloc: nincs memóriaszivargas a programban.
* Osszes foglalas: 32 blokk, 3204 bajt.
*****
```

További bemenetek

A további tesztelés során egyetlen recepteket tartalmazó fájlt használtam, miközben az összetevők helyét, illetve a részlegeket tartalmazó fájlt cserélgettem. Míg a debugolás során használt fájlban az emberek számára vonatkozó adatok nem voltak reálisak a manuálisan könnyebben kezelhető távolságadatok érdekében, a továbbiakban reálisabb adatokkal dolgoztam.

Tesztfájl, más bemenet

Használt fájlok: 0_bevasarlokozpont.txt,
0_osszetevok.txt, receptek.txt

Más kiindulási pont

```
c:\Prog\nagyhazi\karacsony_bev.exe
Adjon meg egy receptet es egy kiindulasi pontot!
<RECEPT> <start>: PISKOTA c

BEVASARLOLISTA
- liszt
- cukor
- tojas
-----
Idealis utvonal: c f a b X
```

```
receptek.txt
1 PISKOTA/0000000000000111
2 ISCHLER/1010010100010011
3 MEZESKALACS/0111000010110111
4 ZSERBO/1000010111011111
5 MAKOSBEJGLI/0001011011011111
6 DIOSBEJGLI/1001000011011111
7 MAKOSGUBA/0000001011001011
8 DIOSGUBA/1000000011001011
9 LINZER/0000010000010111
10 KALACS/0000000010111111
11 MEZESKREMES/0001010110111111
12 MEZESPUSEDLI/0111000000110111
13 VAJASKEKSZ/0000000110110111
14 VANILIASKIFLI/1000000010010011
15 MARCIPANOSKEKSZ/0000100010110111
16 LEKVAROSBUKTA/0000010001011111
17 SZUPERRECEPT/1111111111111111
```

Más recept

Más recepthez nem áll rendelkezésre elég összetevő.

Nagyobb bevásárlóközpont, összes összetevő megvan

Használt fájlok: 1_bevasarlokozpont.txt, 1_osszetevok.txt, receptek.txt

Közepes recept (lekváros bukta, 7 összetevő)

Nagy recept (mézeskrémes, 10 összetevő)

```
c:\Prog\nagyhazi\karacsony_bev.exe
Adjon meg egy receptet es egy kiindulasi pontot!
<RECEPT> <start>: LEKVAROSBUKTA f

BEVASARLOLISTA
- liszt
- cukor
- tojas
- tej
- vaj
- eleszto
- lekvar
-----
Idealis utvonal: f l b c g X
```

```
c:\Prog\nagyhazi\karacsony_bev.exe
Adjon meg egy receptet es egy kiindulasi pontot!
<RECEPT> <start>: MEZESKREMES a

BEVASARLOLISTA
- liszt
- cukor
- tojas
- tej
- vaj
- sutopor
- vanilia
- csokolade
- lekvar
- mez
-----
Idealis utvonal: a b c e g h d l f k X
```

Szupeercept (16 összetevő)

```

c:\Prog\nagyhazi\karacsony_bev.exe
Adjon meg egy receptet es egy kiindulasi pontot!
<RECEPT> <start>: SZUPERRECEPT d

BEVASARLOLISTA
- liszt
- cukor
- tojas
- tej
- vaj
- sutopor
- eleszto
- vanilia
- csokolade
- mak
- lekvar
- marcipan
- mez
- fahej
- szegfuszeg
- dio
-----
Idealis utvonal: d h c e b f k l j g a X

```

Ebben az esetben először már érzékelhető várakozási idő van, a programnak 4-5 másodpercre van szüksége a megoldás kiszámításához, de még ez is bőven használható időn belül megoldást talál a problémára.

Még nagyobb bevásárlóközpont, minden összetevő más részlegen

Használt fájlok: 2_bevasarlokozpont.txt, 2_osszetevok.txt, receptek.txt

Közepes recept (kalács, 6 összetevő)

```

c:\Prog\nagyhazi\karacsony_bev.exe
Adjon meg egy receptet es egy kiindulasi pontot!
<RECEPT> <start>: KALACS o

BEVASARLOLISTA
- liszt
- cukor
- tojas
- tej
- vaj
- eleszto
-----
Idealis utvonal: o f l c g p X

```

Nagy recept (zserbó, 10 összetevő)

```

c:\Prog\nagyhazi\karacsony_bev.exe
Adjon meg egy receptet es egy kiindulasi pontot!
<RECEPT> <start>: ZSERBO g

BEVASARLOLISTA
- liszt
- cukor
- tojas
- tej
- vaj
- eleszto
- vanilia
- csokolade
- lekvar
- dio
-----
Idealis utvonal: g h c b n m l f o p X

```

Szupeerceptek (11+ összetevő)

11 összetevő: kb. 5 sec

12 összetevő: kb. 90 sec, perm.bin már 1.63 GB méretű!

13+ összetevő: nem vártam ki

Ilyen esetek azonban igen valószínűtlenek valós bemenet esetén, hiszen egy recept ritkán tartalmaz ennyi összetevőt és még ritkábban vannak azok mind különböző részlegeken.