Praktikumide ja iseseisva töö ülesanded

MT.03.133 Algoritmid ja andmestruktuurid
2103/2014 sügissemestril

Jüri Kiho

Iseseisev töö nr 1. *ajagraafikud*

Ülesanne P3

P3-1. Sõnade generaator. *gen_sõnad4*

P3-2. Bitijärjendite generaator. *gen_bitivekt*

Ülesanne P4. Variantide läbivaatamine. *var_bobid*

Iseseisev töö nr 2. *var_paadid*

Ülesanne P5

P5-1. Rekursiivne generaator-funktsioon. *gen_rekurs*

P5-2. Suluavaldise arvutamine. *mag_suluavaldis*

Ülesanne P6. Sorteerimise järjekorra- ja magasinimeetod. *ajagraafikud*

Iseseisev töö nr 3. *ajagraafikud*

Ülesanne P7. Loendamine. *loe_korduvaid*

Ülesanne P8. Paiskfunktsioonid. *paiskfunktsioonid*

Iseseisev töö nr 4. *paiskfunktsioonid*

KN.txt
KOVvalimised2013_tulemused.txt
prot_PaideTyri32_13km.txt
prot_rm_2013_40ab_lp.txt
prot_Saaremaa40_pj.txt
prot_TartuSygisj_10km.txt
prot_tm_2013_31ab_lp.txt
prot_tm_2013_63ab_lp-1.txt

Ülesanne P9. Etüüdid puul. *puud; puu.txt*
Ülesanne P10. Etüüdid kahendotsimispuul. *kahendotsimispuud*
Iseseisev töö nr 5. *kop_AVL*;
rikkega, võte 1: *kop1.txt* -- puu sees, *kop3.txt* -- rike juurtipus,
kop4.txt -- rikkega tipul ainult 1 alluv, *kop5.txt* -- puudub "keskmine“,
kop6.txt -- väike
rikkega, topeltvõte: *kop1_2.txt* -- puu sees,
kop4_2.txt -- rikkega tipul (juur) ainult 1 alluv, *kop6_2.txt* -- väike
Iseseisev töö nr 6. *robotiralliKK*
Ülesanne P13. Etüüdid graafil. *graafid*; *graafTest13.txt*
Ülesanne P14
Ülesanne P14-1. Tee graafis (sügavuti). (*sügavuti_teed*)
Ülesanne P14-2. Teed graafis (sügavuti). *sügavuti_teed; graafTeed.txt, graafTeedSuur.txt*
Iseseisev töö nr 7. *Dijkstra, DijkstraKK; graafTeed.txt, graafTeedSuur.txt*
Ülesanne P15. Minimaalne toes
Iseseisev töö nr 8. *kruskal_reis; linnade_graaf_kaugustega.txt, graafTest15+.txt*

Iseseisev töö nr 1

1. Valida välja, realiseerida ja testida läbi kaks aeglast, ruutkeerukusega (keskmise ajalise keerukuse hinnanguga $O(n^2)$) algoritmi järjendi elementide ümberpaigutamiseks mittekahanevasse järjekorda. Sellisteks algoritmideks on näiteks mullimeetod (*bubble sort*), valikumeetod (*selection sort*), pistemeetod (*insertion sort*) jt. Testimisel mitte unustada nn äärejuhte (järjendid pikkusega 0, 1 või 2; ette antakse sorditud järjend, või siis vastupidises suunas sorditud järjend).
2. Valida välja, realiseerida ja testida läbi üks kiire, keskmise ajalise keerukuse hinnanguga $O(n \log n)$ algoritm järjendi elementide ümberpaigutamiseks mittekahanevasse järjekorda. Sellisteks algoritmideks on näiteks kiirmeetod (*quick sort*), ühildusmeetod (*merge sort*), Shelli meetod (*Shell sort*) jt.Testimisel mitte unustada nn äärejuhte!
3. Koostada sortimismeetodite tööaegade graafikute joonistamise programm (graafik näitab tööaja kasvu vastavalt sorditavate järjendite pikkuste kasvule). Rakendada seda programmi nelja sorteerimismeetodi graafiku kujutamiseks (ühel joonisel):
valitud kolm meetodit + süsteemne sortimismeetod, nt Pythoni korral funktsioon *sorted*.
   3a. Sorteeritavad on juhujärjendid.
   3b. Sorteeritavad on sellised eriomadusega järjendid, mis vähemalt ühe aeglase meetodi jaoks on “soodsad”.

Soovitus. Sorteerimismeetodite algoritme leiab kas Internetist või õpikust või loengumaterjalidest.
Vt ka *AAbitsPy* ja http://www.sorting-algorithms.com/ .

Esitada: sortimismeetodite tööaegade graafikute joonistamise programm (3.), mis sisaldab valitud kolme algoritmi realisatsioone ja võimaldab vaadelda variante 3a ning 3b.

Ülesanne P3

P3-1. Sõnade generaator

Kirjutada funktsioon, mis väljastab ekraanile kõik neljatähelised sõnad kujul $\alpha\beta\gamma\gamma$, kus $\alpha \in \{k, l, m, n, p, r, s, t\}$, $\gamma \in \{k, l, m, n, p, r, s, t\}$, $\beta \in \{a, e, i, o, u\}$

Kirjutada vastav generaator-funktsioon (ekraanile väljastamise asemel annab välja järjekordse sõna). Testida.

P3-2. Bitijärjendite generaator

Kirjutada funktsioon, mis väljastab ekraanile kõik bitijärjendid, mille pikkus on $n > 0$ ja milles esineb parajasti k ühte, $0 < k <= n$.

Testida seda funktsiooni, näiteks

1) $n = 5$ ja $k = 3$ korral,
2) $n = 30$ ja $k = 2$ korral.

Kirjutada vastav generaator-funktsioon (ekraanile väljastamise asemel annab välja järjekordse nõutud bitivektori). Testida.

Ülesanne P4

Variantide läbivaatamine

Kirjutada programm järgmise probleemi lahendamiseks, võib võtta $n = 7$.

Probleem: *n*-liikmelisest treeningrühmast (n=10 ± 2) tuleb eelolevaks võistluseks välja valida neljase bobi meeskond. Treeningrühma iga liikme kohta on antud: nimi, kaal ja treenituse tase (reiting) naturaalarvuna.
Valikukriteeriumid bobi meeskonna jaoks:

- kogukaal <= 325 kg;
- reitingute summa on võimalikult suur.

A. Üks valik (kui võimalik).
B. Kõik võimalikud valikud.

Iseseisev töö nr 2

Probleem

Turismifirma korraldab paadisõite ranniku saarestikus.
Kasutada on 3 paati: 7 reisijat, kokku kuni 500 kg,
. 4 reisijat, kokku kuni 350 kg,
. 4 reisijat, kokku kuni 350 kg.
Reisijate arv: 15;
. reisijad on numbritega 0 ... 14
. ja kaaludega (näiteks) 95, 85, 60, 90, 116, 75, 85, 60, 60, 65, 95, 85, 60, 70, 75.
Erisoovid: teatud paarid peavad saama ühte paati, nt paarid (1,5), (4,11), (7,13), (0,10).

Leida kõik võimalikud reisijate paigutusviisid, näiteks üks selline oleks:
. I paat: reisijad (3, 7, 8, 9, 12, 13, 14) kaaludega [90, 60, 60, 65, 60, 70, 75],
kokku 480 kg
. II paat: reisijad (2, 4, 6, 11) kaaludega [60, 116, 85, 85], kokku 346 kg
. III paat: reisijad (0, 1, 5, 10) kaaludega [95, 85, 75, 95], kokku 350 kg

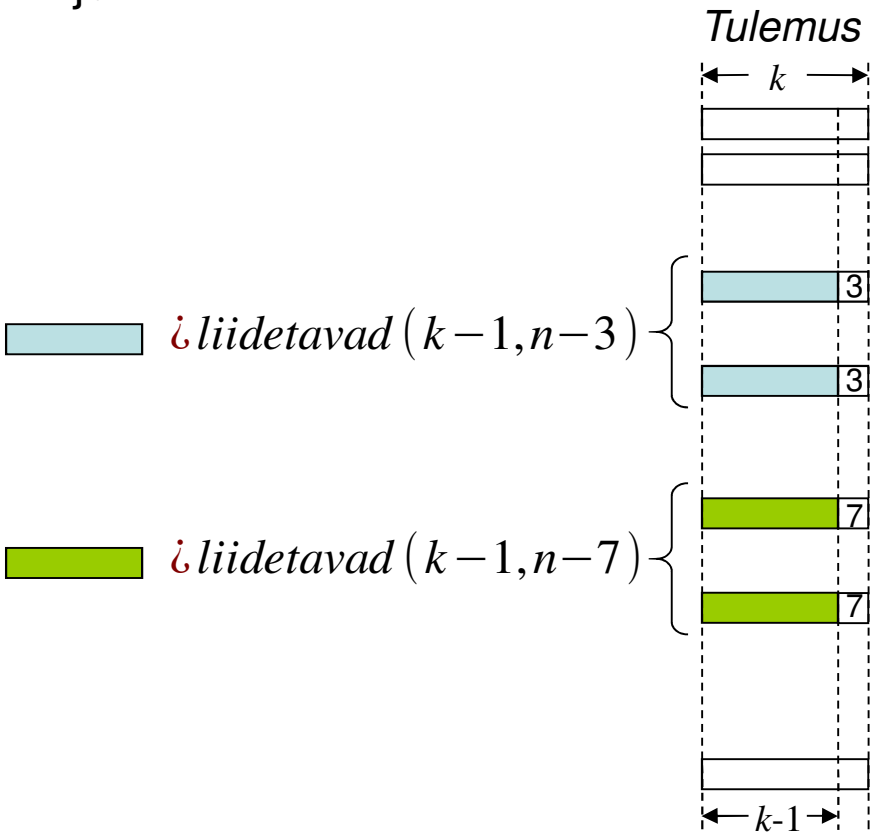
Kirjutada programm selle probleemi lahendamiseks. Testimisel varieerida andmeid.

Ülesanne P5

P5-1. Rekursiivne generaator-funktsioon:

```
def liidetavad(k, n)
    #Generaator-funktsioon, rekursiivne
    # Antud (algatamisel): positiivsed täisarvud k ja n
    # Tulemus (pöördumisel): antakse välja järjekordne arvukomplekt,
    #                        milles on k mittenegatiivset arvu, mille summa on n
```

Vihje:



(Ülesanne P5)

P5-2. Suluavaldise arvutamine.
Kirjutada programm järgmise ülesande lahendamiseks.

Antud on mingi suluavaldis (sõnena), näiteks

```
(((a+b)*(c-d*k)+2.5)/(pi*(1-e)))
```

Kogu avaldis on ka sulgudes.

Väljastada arvutuseeskiri, mis näitab millises järjekorras tuleb (võiks) arvutada selles esinevate suluavaldiste väärtused, näiteks:

```
Arvutada:
S1 := a+b
S2 := c-d*k
S3 := S1*S2+2.5
S4 := 1-e
S5 := pi*S4
S6 := S3/S5
```

Programm peab kasutama magasini.

Edasijõudnud: tuvastada ka sulustatuse ebakorrektsus ("*invalid syntax*").

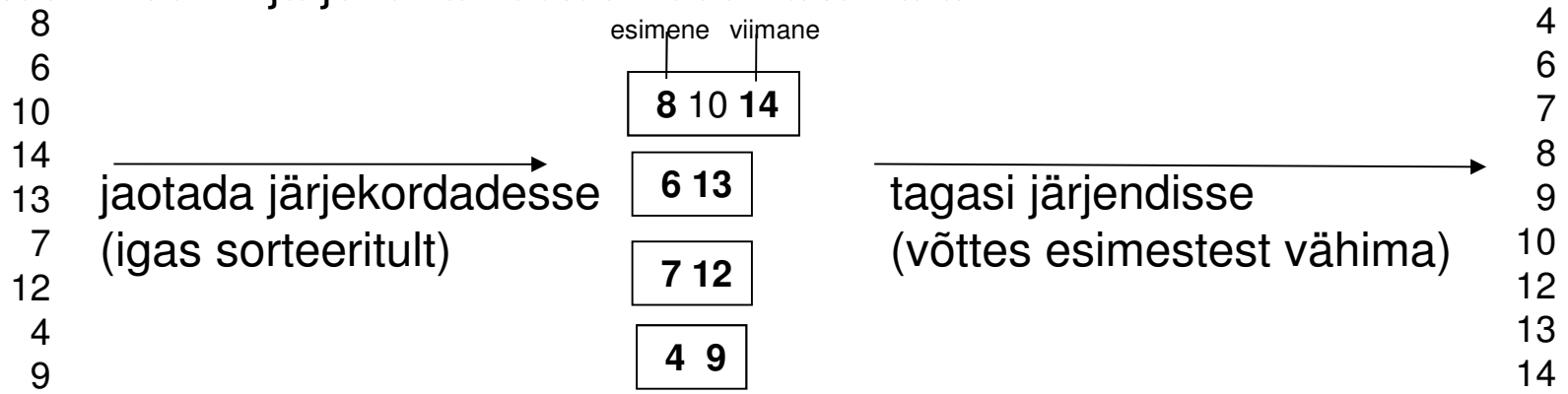
Ülesanne P6

Sorteerimise järjekorra- ja magasinimeetod

P6-1. Koostada sorteerimisfunktsioon, mis põhineb magasinidesse jaotamisel.
I osa: paigutada järjendi elemendid magasinidesse, igas mittekasvavalt, nt
[**7, 4, 5, 3, 1, 2, 8, 6, 4**] → [7, 4, 3, **1**], [5, **2**], [8, 6, **4**]
II osa: korjata magasinidest tulemusjärjend.

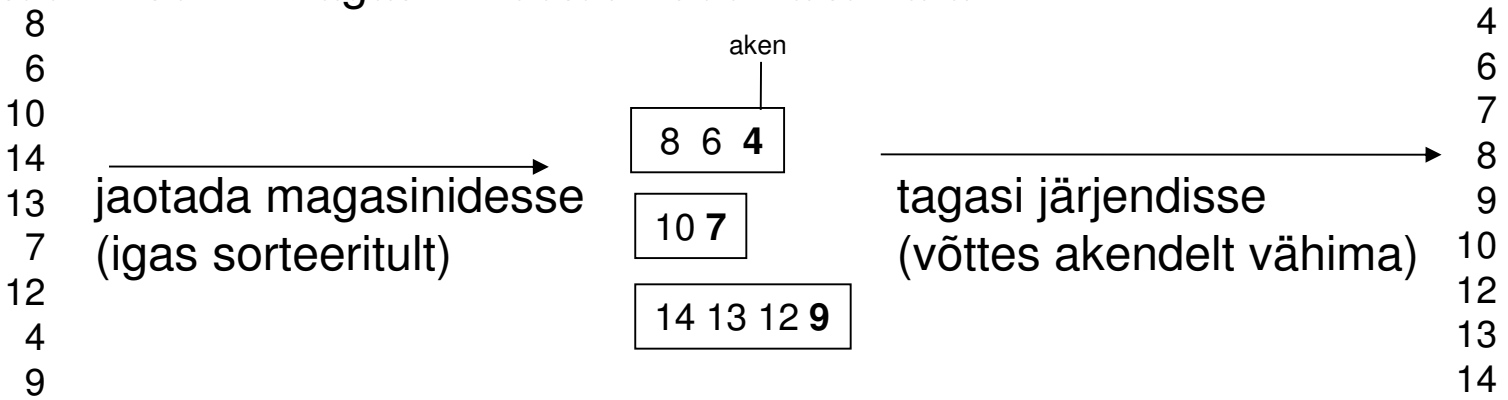
P6-2. Koostada sorteerimisfunktsioon, mis põhineb järjekordadesse jaotamisel.
I osa: paigutada järjendi elemendid järjekordadesse,
igas mittekahanevalt (suunas esimene->viimane), nt
[**1, -7, 4, 5, 8, 3, 6, -2, 3, 4**] → [**1**, 4, 5, **8**] [**-7**, 3, **6**] [**-2**, 3, **4**]
II osa: korjata järjekordadest tulemusjärjend.

Sorteerimise nn järjekorrameetod. Idee näite varal:



Millisest klassist on selle algoritmi keerukus?
Millise järjendi korral töötab kõige kiiremini / kõige aeglasemalt?

Sorteerimise nn magasinimeetod. Idee näite varal:



Millisest klassist on selle algoritmi keerukus?
Millise järjendi korral töötab kõige kiiremini / kõige aeglasemalt??

Iseseisev töö nr 3

A .Koostada sorteerimisfunktsioon, mis põhineb järjekordadesse jaotamisel, vt P6-2. (Ei ole vaja eraldi esitada.)
Koostada ja esitada programm sorteerimismeetodite katselis-graafiliseks uurimiseks. Uuritavateks olgu järjekorrameetod, üks ruutkeerukusega ja üks kiirem (O(n log$n$)) sorteerimismeetod. Programm peab demonstreerima tööaegade graafikuid (1) juhujärjendite korral ja (2) järjekorrameetodile soodsate järjendite korral.

B. Koostada sorteerimisfunktsioon, mis põhineb magasinidesse jaotamisel vt P6-1. (Ei ole vaja eraldi esitada.)
Koostada ja esitada programm sorteerimismeetodite katselis-graafiliseks uurimiseks. Uuritavateks olgu magasinimeetod, üks ruutkeerukusega ja üks kiirem (O(n log$n$)) sorteerimismeetod. Programm peab demonstreerima tööaegade graafikuid (1) juhujärjendite korral ja (2) magasinimeetodile soodsate järjendite korral.

Ülesanne P7

Loendamine

```
 # Algoritmid ja andmestruktuurid
 # Ülesanne P7-1: Korduvate väärtuste loendamine
 # Autor: Ülli Õpilane
 __version__ = "3.2"
def print_kordumised(a):
        # Antud: järjend (listina) a
        # Tulemus: a korduvate elementide korral prinditakse read kujul
        #.              " <väärtus> esineb <esinemisi> korda
        # Näiteks, kui a = [9, 0, -2, 9, 8, 6, 0, 8, 9, -1], siis prinditakse
                        # ------------------------------------
                        # Korduvad elemendid jarjendis
                        # [9, 0, -2, 9, 8, 6, 0, 8, 9, -1]:
                        #         9 esineb 3 korda
                        #         0 esineb 2 korda
                        #         8 esineb 2 korda
                        #------------------------------------

        print("------------------------------------")
        print("Korduvad elemendid jarjendis\n", a, ':')

        ...
        print("------------------------------------")
# Test
a = [9, 0, -2, 9, 8, 6, 0, 8, 9, -1]
print_kordumised(a)
n = 15
# teha täisarvude juhujärjend b pikkusega n,
import random
b = [random.randint(10, 15) for i in range(n)]
print_kordumised(b)
```

Ülesanne P8

Paiskfunktsioonid

Ülesanne P8-2. Koostada programm erinevate (nt räsifunktsioonil põhinevate) paiskfunktsioonide katsetamiseks mitmesuguste andmefailide korral.

Väljastab statistikat, nt:

paiskfunktsiooni iseloomustus ---------

```
Fail: KN.txt
M = 33
h: (räsi%21474836471)%M  JK3
        2 kirjet yhte slotti 3 korda
        3 kirjet yhte slotti 1 kord
Kirjete arv: 22
Unikaalseid h vaartusi: 13 (59%)
```

Iseseisev töö nr 4

I. Ülesanne P8-2. Koostada ja esitada programm.

Programmis peab olema defineeritud 3 paiskfunktsiooni:

paiskfunktsioon 1: lihtsaim (*räsi*(*võti*) % *M*)

paiskfunktsioon 2: muu, loengust või internetist/kirjandusest

paiskfunktsioon 3: originaalne, omaloominguline

II. Täita ja esitada tabel, üldkuju:

M = [*kirjete arv* * 1.5]

Kirjefaile:

- KN.txt
- KOVvalimised2013_tulemused.txt
- prot_PaideTyri32_13km.txt
- prot_rm_2013_40ab_lp.txt
- prot_Saaremaa40_pj.txt
- prot_TartuSygisj_10km.txt
- prot_tm_2013_31ab_lp.txt
- prot_tm_2013_63ab_lp-1.txt

Nr	*Faili nimi*	*Kirjete arv*	*M*	*paiskfn* 1	*paiskfn* 2	*paiskfn* 3
	<kirjefail 1>			<p11>	<p12>	<p13>
	<kirjefail 2>			<p21>		
	<kirjefail 3>					
	<kirjefail 4>					
	<kirjefail 5>					
	<kirjefail 6>					<p63>
		Keskmiselt:		<kp1>	<kp2>	<kp3>

p*ij* = *j*-nda paiskfunktsiooni unikaalsete väärtuste protsent *i*-nda kirjefaili korral

Ülesanne P9

II. Etüüdid puul

Nr	Antud	Tulemus	Kasutab	Rek
II-1	Puu ja selle tipp	Tagastatakse antud tipust algava (alam)puu kõrgus	F-n *naabertipud* klassist *Graaf*	+
II-2	Puu ja selle tipp	Tagastatakse antud tipu järglaste (vahetute ja kaugemate) koguarv	F-n *naabertipud* klassist *Graaf*	+
II-3	Puu ja selle tipp	Väljastatud konsoolile antud tipust algava (alam)puu vasak suluesitus [Kiho 2003, lk 43]	F-n *aste* klassist *Graaf* F-n *märgend* klassist *Tipp* F-n *naabertipud* klassist *Graaf*	+

P9. Realiseerida ja testida II-1. II-2, II-3

Ülesanne P10

I. Etüüdid kahendotsimispuul

I-0. Kontrollida, kas antud kahendpuu on kahendotsimispuu

Nr	Antud kahendotsimispuu ja selle tipp – alampuu juur …	Tulemus	Rek
I-1	… ning lisatav arv	Antud arv lisatud kahendotsimispuusse (uueks tipuks)	+
I-2	... ning otsitav arv	Tagastatakse tipp, milles on antud arv, ja selle tipu ülemus (või *None*)	+
I-3	… ning eemaldatav tipp	Antud tipp eemaldatud kahendotsimispuust	+
I-4	… ning eemaldatav arv	Antud arvu sisaldav tipp eemaldatud kahendotsimispuust	-
I-5		Tagastatakse *True*, kui alampuu üheski tipus alluvate kõrgus ei erine rohkem kui 1 võrra (st alam-kahend-puu on tasakaalus), muidu *False*	+

P10. Realiseerida ja testida I-0. I-1, I-2, I-3, I-4, I-5

Iseseisev töö nr 5

Esitada viimistletud programm *KOP_AVL* lähtudes toorikust *KOP_AVL_mall*

Iseseisev töö nr 6

Programmeerida ümber programm *Robotiralli*, esitades andmestruktuuri kahendkuhjana.
Viimane peab sisaldama funktsioone *vasak*, *parem*, *viia_alla*, *võtta_vähim*

Etüüdid graafil

Ülesanne P13

Nr	Antud	Tulemus	Kasutab	Rek
III-1	Graaf ja selle tipp *t*	Nende tippude arv, mis on saavutatavad lähtudes tipust *t*	F-n *seadaVäli* klassist *Tipp* F-n *väli* klassist *Tipp*	+
III-2	Graaf ja selle tipp *t*	Tagastatakse *True*, kui antud graaf kujutab puud juurtipuga *t*, *False* vastasel korral.	III-1 F-n *sisendaste* klassist Graaf	- (+)
III-3	Graaf	Antud graaf teisendatud sümmeetriliseks: graafile lisatud puudunud vastandkaared. Kaare (*t*1, *t*2) vastandkaar on (*t*2, *t*1); siin: sama nimega.	Väli *kaared* klassis *Graaf* F-n *otsidaKaar* klassis *Graaf* F-n *lisadaKaar* klassis *Graaf* Väli *algusTipp* klassis *Kaar* Väli *lppTipp* klassis *Kaar* Väli *nimi* klassis *Kaar* Kaare konstruktor: *Kaar*()	-
III-4	Sümmeetriline graaf	Tagastatakse *True*, kui selles graafis leidub tsükleid, *False* vastasel korral. (Eemaldada astmega 0 või 1 tippe; kui kõik sai eemaldada, siis tsükleid ei olnud.)	Väli *tipud* klassis *Graaf* F-n *aste* klassist *Graaf* F-n *eemaldadaTipp* klassist *Graaf* Testimisel: f-n eelmisest etüüdist	-
III-5	Sümmeetriline graaf	Tagastatakse *True*, kui see graaf on sidus, *False* vastasel korral. (Kui ühest suvalisest tipust lähtudes on saavutatavad kõik teised tipud, siis see graaf on sidus.)	III-1	-

P13. Realiseerida ja testida III-1. III-2, III-3, III-4

Ülesanne P14

Ülesanne P14-1. Tee graafis (sügavuti)

Programmeerida programmi *SygavutiTee* (vt Moodle/Praktikum_14) testimise osa.
Näitegraafiks võiks olla *graafTeed.txt,* milles otsida nt teed tipust nr 1 tippu nr 12.

Graafi kaare *k* pikkuseks on arv, mis antud sõnena väljal *k.nimi.*

Ülesanne P14-2. Teed graafis (sügavuti)

Lähtudes programmist *SygavutiTee* koostada programm (*SygavutiTeed*),
mis leiab kõik elementaarteed graafis antud tipust teise antud tippu ja tuvastab ka nendest lühima (st väikseima kaarepikkuste summaga tee).

Graafi kaare *k* pikkuseks on arv, mis antud sõnena väljal *k.nimi.*

Iseseisev töö nr 7

Realiseerida Dijkstra algoritm [Kiho 2003, lk 98,100,101] . Testimisel kontrollida tulemusi, võrreldes neid ülesandes P14-2 koostatud lühima tee sügavuti leidmise programmi tulemustega.

Mittekohustuslik: eelistusjärjekord *Q* realiseerida tippude (pöörd)kahendkuhjana.
modifitseerides vahendeid oma eelmisest kodutööst (nr 6).

Ülesanne P15

Minimaalne toes

P15. Programmeerida funktsioonide (Galler-Fischeri klassikäitlus, *Kruskal_toes*) sisud programmi toorikus *P15_Kruskal*.

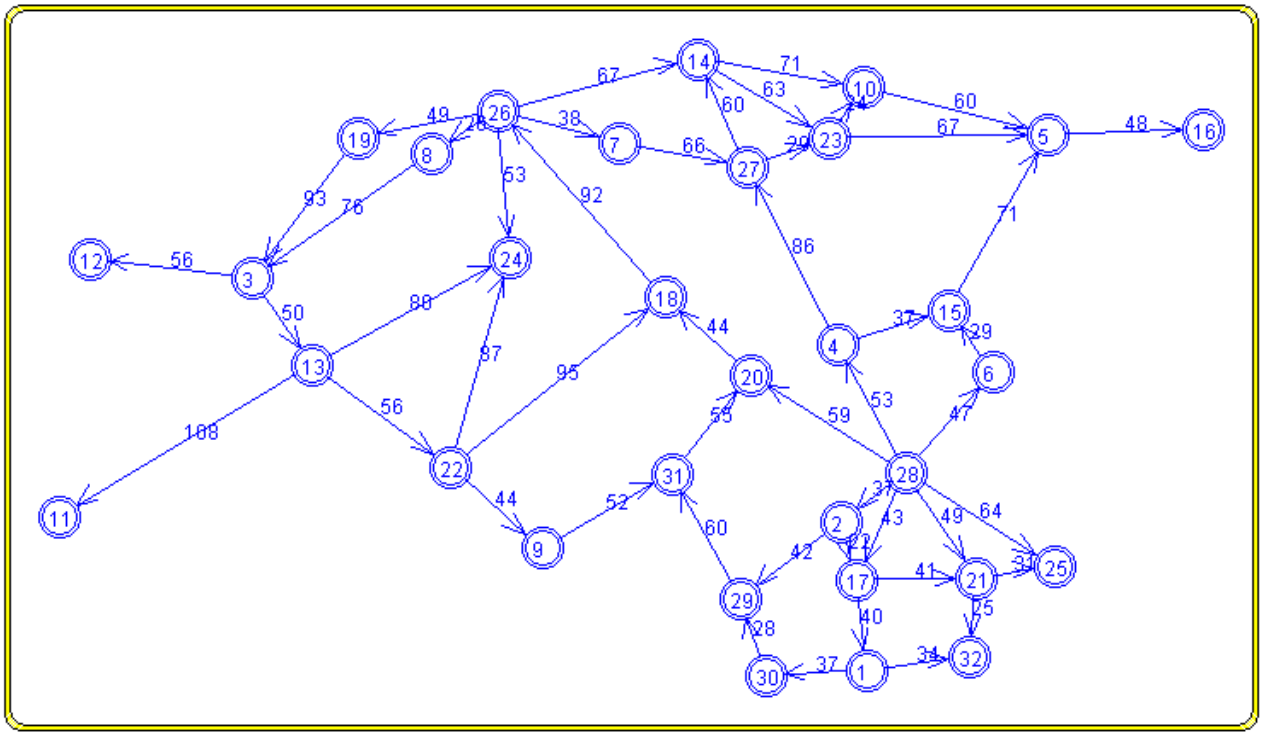
Iseseisev töö nr 8

Programm järgmise ülesande lahendamiseks.
Asulate plaan on antud sümmeetrilise graafina, milles tipud esindavad asulaid ja kaared nendevahelisi ühendusteid, vt järgmine slaid. Tipu märgendiks on asula nimi, kaarenimeks – ühendustee pikkus (arv sõnena).
Koostada rännaku eeskiri kõigi asulate külastamiseks mööda selle graafi minimaalse toese servi, lähtudes antud lähteasulast. Vt ülejärgmine slaid – eeskiri juhul, kui lähteasulaks on “viljandi”. (Noolega märgitud tippu on jõutud uuesti, st tagasiteel.)

Programmi struktuur (vt *KruskalReisMall*):

- Leida graafi minimaalne toes Kruskali meetodil
- Fikseerida lähtetipp (lähteasula)
- Printida rännaku eeskiri

C:_ISIKLIK\AlgPython\workspace2013\linnade_graaf_kaugustega.txt



TABEL
1) antsla 37-->30 34-->32
2) elva 42-->29 22-->17
3) haapsalu 56-->12 50-->13
4) jogeva 86-->27 37-->15
5) johvi 48-->16
6) kallaste 29-->15
7) kehra 66-->27
8) keila 76-->3
9) kilinginomme 52-->31
10) kunda 60-->5
11) kuressaare
12) kärdla
13) lihula 108-->11 56-->22 80-->24
14) loksa 71-->10 63-->23
15) mustvee 71-->5
16) narva
17) otepää 40-->1 41-->21
18) paide 92-->26
19) paldiski 93-->3
20) poltsamaa 44-->18
21) polva 31-->25 25-->32
22) pärnu 87-->24 95-->18 44-->9
23) rakvere 24-->10 67-->5
24) rapla
25) räpina
26) tallinn 67-->14 38-->7 26-->8 49--
27) tapa 29-->23 60-->14
28) tartu 47-->6 53-->4 37-->2 43-->17
29) torva 60-->31
30) valga 28-->29
31) viljandi 55-->20
32) voru

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremklõps väljaspool graafi ala.
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasakuga. Eemaldamine: parem klõps valitud tipul (2 x parem klõps)
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algustipp, seejärel lõpptipp.
GRAAF Ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.

FAIL: C:_ISIKLIK\AlgPython\workspace2013\linnade_graaf_kaugustega.txt Värskendamine: pidev.

31 viljandi
20 poltsamaa
18 paide
---> 20 poltsamaa
28 tartu
6 kallaste
15 mustvee
5 johvi
16 narva
---> 5 johvi
10 kunda
23 rakvere
27 tapa
14 loksa
---> 27 tapa
7 kehra
26 tallinn
8 keila
---> 26 tallinn
19 paldiski
---> 26 tallinn
24 rapla
---> 26 tallinn
---> 7 kehra
---> 27 tapa
---> 23 rakvere
---> 10 kunda
---> 5 johvi
---> 15 mustvee
4 jogeva
---> 15 mustvee
---> 6 kallaste
---> 28 tartu

2 elva
17 otepää
1 antsla
30 valga
29 torva
---> 30 valga
---> 1 antsla
32 voru
21 polva
25 räpina
---> 21 polva
---> 32 voru
---> 1 antsla
---> 17 otepää
---> 2 elva
---> 28 tartu
---> 20 poltsamaa
---> 31 viljandi
9 kilinginomme
22 pärnu
13 lihula
11 kuressaare
---> 13 lihula
3 haapsalu
12 kärdla
---> 3 haapsalu
---> 13 lihula
---> 22 pärnu
---> 9 kilinginomme
---> 31 viljandi