Datastructuren en graafalgoritmen

tuyaux.winak.be/index.php/Datastructuren_en_graafalgoritmen

Datastructuren en graafalgoritmen	
Richting	Informatica,Wiskunde
Jaar	3BINF,Bachelor Wiskunde Keuzevakken

Bespreking

Dit vak wordt gegeven door Benny Van Houdt. Een sympathieke prof die de lessen goed uitlegd, toch is het niet altijd even gemakkelijk op te blijven letten omdat de leerstof heel abstract is. In de les komen ook veel tekeningen op bord, iets wat de cursus wel mist. Verder is het een duidelijk uitgelegde dunne (60 bladzijden) cursus, die heel volledig is. Tijdens het jaar worden ook oefeningen opgegeven waarmee je punten kan winnen voor je examen, je kan deze oefeningen best maken aangezien de moeilijkheid van het examen wordt aangepast op het gemiddeld aantal punten op de oefeningen tijdens het jaar. Op het examen zelf kun je kiezen of je open of gesloten boek kiest. Beide exames hebben 4 dezelfde oefeningen en daarbij krijg je nog 3 extra oefeningen (open boek) of 3 theorievragen (gesloten boek). Als je de oefeningen in de cursus zonder teveel moeite kunt oplossen is het open boek zeker geen slechte keuze. Als je liever wat zekerheid wilt kan je voor het gesloten boek examen gaan.

Puntenverdeling

Examen op 20 punten.

Examenvragen

Academiejaar 2021 - 2022 (1ste zittijd)

Bestand:DSnGA 2022 closed.pdf

Academiejaar 2020 - 2021 (1ste zittijd)

Gesloten Boek Examen

Theorie

1. **Graaf Searching (3):** Leg uit wat we bedoelen meet een topological sort van een acyclische graaf GG. Geef een snel algoritme voor de topological sort en bewijs de correctheid ervan.
2. **Flow Netwerken (4):** Geef een definitie van een flow ff. Wat is het residual netwerk GfGf ten opzichte van een flow ff? Hoe kunnen we GfGf gebruiken om de flow ff te verhogen (indien mogelijk)? Bewijs dit laatste.

3. **Kortste Paden (3):** Hoe werkt de *Relax* operatie? Toon aan dat wanneer we een reeks *Relax* operaties uitvoeren d(v)d(v) steeds minstens even groot is als δ(s,v)δ(s,v) wanneer we de dd-waardes initialiseren met *Initialize-Single-Source*

Oefeningen

1. **Graaf Searching (2):** Geef twee grafen G1=(V1,E1)G1=(V1,E1) en G2=(V2,E2)G2=(V2,E2) zodat één
   G1

   G1

   precies k1k1 SCCs bevat, twee
   G2

   G2

   precies k2k2 SCCs heeft en drie: er bestaat u1,u2∈V1u1,u2∈V1 en v1,v2∈V2v1,v2∈V2 zodat G=(V1∪V2,E1∪E2∪{(u1,v1),(v2,u2)})G=(V1∪V2,E1∪E2∪{(u1,v1),(v2,u2)}) slechts één SCC bevat.
2. **Disjoint Sets (3):** Stel dat we enkel *Path Compression* gebruiken en geen *Union-By-Rank*. Geef dan een reeks n=10n=10 *MakeSet*, 9 *Union* en 10 *FindSet* operaties zodat het aantal parent pointers dat je moet aanpassen tijdens de *FindSet* operaties zo groot mogelijk is. De *FindSet* operaties die deel uitmaken van de *Union* tellen we niet mee. Teken je datastructuur voor elke *FindSet* en nummer je knopen zodat het duidelijk is welke pointers aangepast zijn.
3. **Opspannende Bomen (2):** Geef een graaf GG met 4 knopen en 6 edges met gewichten 1, 2, 3, 4, 5 en 6 zodat het gewicht van de opspannende boom met minimaal gewicht zo groot mogelijk is.
4. **Fibonacci Heaps (3):** Stel dat we geen markeringen gebruiken en als potentiaal functie Φ(H)=t(H)Φ(H)=t(H). Wat is dan de geamortiseerde kost van een *DeleteMin* en *DecreaseKey* operatie? Waarom is de complexiteit minder goed dan markeringen? Leg Uit.

Open Boek Examen

1. Graaf Searching (3): Op hoeveel verschillende manieren kan je de knopen van de graaf G=(V,E)G=(V,E) topologisch sorteren indien V={s,u1,u2,v1,v2,v3,t}V={s,u1,u2,v1,v2,v3,t} en E={(s,u1),(u1,u2),(u2,t),(s,v1),(v1,v2),(v2,v3),(v3,t)}E={(s,u1),(u1,u2),(u2,t),(s,v1),(v1,v2),(v2,v3),(v3,t)}? Leg je antwoord duidelijk uit.
2. Graaf Searching (2): Geef twee grafen G1=(V1,E1)G1=(V1,E1) en G2=(V2,E2)G2=(V2,E2) zodat:
   1. G1G1 precies k1k1 SCCs heeft,
   2. G2G2 precies k2k2 SCCs heeft,
   3. er bestaan u1,u2∈V1 en v1,v2∈V2u1,u2∈V1 en v1,v2∈V2 zodat G=(V1∪V2,E1∪E2∪{(u1,v1),(v2,u2)})G=(V1∪V2,E1∪E2∪{(u1,v1),(v2,u2)}) slechts één SCC bevat.
3. Flow Netwerken (3): Wat is het maximum aantal inimale cuts in een flow network bestaande uit nn knopen? Geef een voorbeeld van een flow network waarvoor het aantal minimale cuts gelijk is aan je antwoord.

4. Disjoint Sets (3): Stel dat we enkel Path Compression gebruiken en geen Union-By-Rank. Geef dan een reeks van 10 MakeSet, 9 Union en 10 FindSet operaties zodat het aantal pointers dat je moet aanpassen tijdens de FindSet operaties zo groot mogelijk is. De FindSet operaties die deel uitmaken van de Union tellen we niet mee. Teken je datastructuur voor elke FindSet en nummer je knopen zodat het duidelijk is welke pointers aangepast zijn.
5. Opspannende Bomen (2): Geef een Graaf GG met 4 knopen en 6 edges met gewichten 1,2,3,4,5 en 6 zodat het gewicht van de opspannende boom met minimaal gewicht zo groot mogelijk is.
6. Fibonacci Heaps (4): Stel dat we geen markeringen gebruiken en als potentiaal functie $\Phi(H)=t(H)\Phi(H)=t(H)$. Wat is dan de geamortiseerde kost van een DeleteMin en DecreaseKey operatie? Waarom is de complexiteit minder goed dan met markeringen? Leg uit.
7. Kortste Paden (3): Geef een snel algoritme dat voor elke knoop $v \in Vv \in V$ een zo kort mogelijke cycle CvCv zoekt zodat vv gelegen is op deze cycle (waarbij de lengte van de cycle gelijk is aan het aantal edges die deel uitmaken van de cycle). Wat is de complexiteit van je algoritme?

Academiejaar 2018 - 2019 (2de zittijd)

Gesloten Boek Examen

Theorie

1. GRAAF SEARCHING (3): Het DFS algoritme deelt alle edges op in back, forward en cross edges. Geef een definitie voor elk type edge evenals een voorbeeldje. Geef aan hoe het DFS algoritme weet of een edge een back, forward of cross edge is. Welke types edges treffen we enkel aan in ongerichte grafen en hoe kunnen we deze types gebruiken om na te gaan of een gerichte graaf acyclisch is (je mag beide vragen beantwoorden zonder bewijs).
2. OPSPANNENDE BOMEN (4): Leg uit hoe het algoritme van Kruskal werkt en bewijs dat het een opspannende boom met minimaal gewicht als output geeft. Bespreek eveneens de tijdscomplexiteit van dit algoritme (in voldoende detail).
3. FIBONACCI HEAPS (3): Leg uit hoe de delete-min operatie in een Fibonacci heap werkt. Bepaal de geamortizeerde kost van deze operatie.

Oefeningen

1. GRAAF SEARCHING (3): Geef een algoritme met tijdscomplexiteit O(|V|+|E|) om het langste pad te vinden in een gerichte acyclische graaf. Beschrijf met woorden hoe en waarom je algoritme werkt (dus geen pseudocode).
2. FLOW NETWERKEN (3): De minimale cut in een flow netwerk is niet noodzakelijk uniek. Beschouw een netwerk dat enkel gehele capaciteiten bevat en geef aan hoe we een minimale cut (S, T) kunnen bepalen waarvoor het aantal edges dat loopt van S naar T zo klein mogelijk is (Hint: verhoog de capaciteit van alle edges met een constante)

3. DISJOINT SET DATA STRUCTUREN (4=3+1): Veronderstel dat we een disjoint-sets datastructuur implementeren op basis van een collectie bomen en gebruik maken van path compression en union-by-rank. Geef een voorbeeld van een reeks MAKESET, UNION en FINDSET operaties, startende van een lege datastructuur, zodat een boom TkTk kind wordt van een boom TpTp, hoewel het langste pad in TkTk langer is dan het langste pad in de boom TpTp? Teken het resultaat na elke operatie. Is het gebruik van path compression noodzakelijk om zo een voorbeeld te maken?

Open Boek Examen

1. GRAAF SEARCHING (3): Geef een algoritme met tijdscomplexiteit O(|V|+|E|) om het langste pad te vinden in een gerichte acyclische graaf. Beschrijf met woorden hoe en waarom je algoritme werkt (dus geen pseudocode).
2. GRAAF SEARCHING (3): Gegeven een gerichte graaf G waarbij de knopen zijn genummerd van 1 tot |V|. Laat min(u) de knoop zijn met het kleinste nummer die bereikbaar is vanuit u. Geef een algoritme met tijdscomplexiteit O(|V|+|E|) dat min(u) bepaalt voor alle knopen u∈Vu∈V. Beschrijf met woorden hoe en waarom je algoritme werkt (dus geen pseudo code). (Hint: de reversed graaf.)
3. FLOW NETWERKEN (3): De minimale cut in een flow netwerk is niet noodzakelijk uniek. Beschouw een netwerk dat enkel gehele capaciteiten bevat en geef aan hoe we een minimale cut(S,T) kunnen bepalen waarvoor het aantal edges dat loopt van S naar T zo klein mogelijk is. (Hint: verhoog de capaciteit van alle edges met een constante.)
4. DISJOINT SET DATA STRUCTURES (4=3+1): Veronderstel dat we een disjoint-sets datastructuur implementeren op basis van een collectie bomen en gebruik maken van path compression en union-by-rank. Geef een voorbeeld van een reeks MAKESET, UNION en FINDSET operaties, startende van een lege datastructuur, zodat een boom TkTk kind wordt van een boom TpTp, hoewel het langste pad in TkTk langer is dan het langste pad in de boom TpTp? Teken het resultaat na elke operatie. Is het gebruik van path compression noodzakelijk om zo een voorbeeld te maken?
5. OPSPANNENDE BOMEN (4=2+2): We stellen dat een edge (u,v) een lichte edge is indien er een cut (S, V\S) bestaat zodat u∈S,v∉Su∈S,v∉S en er bestaat geen edge tussen S en V \ S met een gewicht kleiner dan het gewicht van (u,v). Toon aan of volgende uitspraken juist of fout zijn:
    1. Als (u,v) deel uitmaakt van een opspannende boom met minimaal gewich, dan is (u,v) een lichte edge.
    2. De verzameling van alle lichte edges vormt samen een opspannende boom met minimaal gewicht.
6. FIBONACCI HEAPS (3): Hoeveel knopen zitten er minstens in een Fibonacci heap als je weet dat de rootlist bestaat uit drie knopen die respectievelijk 2, 4 en 6 kinderen hebben? Verklaar je antwoord.

Academiejaar 2018 - 2019 (1ste zittijd)

Oefeningen

1. Graaf Searching (3): Geef een voorbeeld van een graaf G = (V, E) met V = {1,...,n} en |E| = n(n+1)/2 -1 zodat BFS(G,1) en DFS(G) dezelfde output geven wanneer alle adjacency lists oplopend gesorteerd zijn. [Hint: start met n klein.]

2. Graaf Searching (3 = 2 + 1): Gegeven een graaf G = (V, E) met SCCs C1,...,CkC1,...,Ck. Laat V(Ci)=u∈V|u∈Ci,E(Ci)=(u,v)∈E|u,v∈Ci,E(Ci,Cj)=(u,v)∈E|u∈Ci,v∈CjV(Ci)=u∈V|u∈Ci,E(Ci)=(u,v)∈E|u,v∈Ci,E(Ci,Cj)=(u,v)∈E|u∈Ci,v∈Cj. Stel dat Gi=(V,Ei)Gi=(V,Ei) dezelfde k SCCs heeft. Wat is de kleinst mogelijke waarde van |Ei||Ei|? Wijzigt je antwoord indien we eveneens eisen dat Ei⊆EEi⊆E? Leg uit.
3. Flow Netwerken (3): Laat f een flow zijn in G = (V, E), f* een maximale flow en Gf(δ)Gf(δ) het residual netwerk van G tov de flow f waarbij we alle edges met cf(u,v)≤δcf(u,v)≤δ hebben verwijderd. Toon aan dat |f∗|≤|f|+δ|E||f∗|≤|f|+δ|E| wanneer er geen pad is van s naar t in Gf(δ)Gf(δ). [Hint: Kies een bepaalde cut (S',T') en maak gebruik van het feit dat |f∗|≤c(S,T)|f∗|≤c(S,T) voor elke cut (S, T).]
4. Flow Netwerken (4 = 1.5 + 1.5 + 1): Gebruik de notatie van de voorgaande oefening. Stel dat alle edges een gehele capaciteit hebben tussen 1 en C. Beschouw de volgende code:

```
f=0; δ=2log2Cδ=2log2C
while δ≥1δ≥1 do
while There exists a path p from s to t in Gf(δ)Gf(δ) do
augment ff with fpfp
end while
δ=δ/2δ=δ/2;
end while
```

Beantwoord volgende vragen en leg uit (gebruik de eigenschap in de voorgaande oefening indien nodig):

1.
    1. Eindigt dit algoritme steeds en is f op het einde een maximale flow?
    2. Hoe vaak voeren we elk van de while lussen maximaal uit?
    3. Wat is de totale complexiteit van dit algoritme?
2. Disjoint Set Data Structures (3 = 2 + 1): Gegeven een graaf G = (V, E). Wat wordt er getest door onderstaand algoritme? Geef voldoende uitleg. Gebruiken we best de linked-list of forest implementatie?

```
for u∈Vu∈V do
MakeSet(u);
end for
Select random e′=(u′,v′)∈E;E′=Ee′e′=(u′,v′)∈E;E′=Ee′.
while E′≠ϕE′≠ϕ do
for e=(u,v)∈E′e=(u,v)∈E′ do
if FindSet(u) = FindSet(v) then
Return False;
else
if FindSet(u) = FindSet(u') then
Union(FindSet(v), FindSet(v')); E' = E'\e;
end if
if FindSet(u) = FindSet(v') then
Union(FindSet(v), FindSet(u')); E' = E'\e;
end if
```

```
if FindSet(v) = FindSet(u') then
Union(FindSet(u), FindSet(v')); E' = E'\e;
end if
if FindSet(v) = FindSet(v') then
Union(FindSet(u), FindSet(u')); E' = E'\e;
end if
end if
end for
end while
Return True
```

1. Opspannende bomen (2): Geef een snel algoritme voor het vinden van een opspannende boom met maximaal gewicht. Leg uit.
2. Fibonacci Heaps (2): Wat is de geamortizeerde kost van de Delete-Min en Decrease-Key operatie wanneer we de potentiaal functie 2t(H)+3m(H) gebruiken? Werk uit.

Academiejaar 2009 - 2010 - 1ste zittijd

1. GRAPH SEARCHING: De square van een gerichte graaf G=(V,E)G=(V,E) is de graaf G2=(V,E2)G2=(V,E2) waarbij (u,w)(u,w) in E2E2 zit als en slechts als er een knoop vv bestaat zodat (u,v)(u,v) en (v,w)(v,w) in EE zitten. Er is dus een edge tussen uu en ww in G2G2 als er een pad was van uu naar ww in GG van lengte twee. Geef een efficiënt algoritme om zowel de incidentie-matrix als adjacency-lijst van G2G2 op te stellen op basis van deze van GG. Bespreek de complexiteit van je oplossing.
2. DISJOINT-SETS FOREST: Stel dat we een array PARTOF van nn elementen gebruiken en PARTOF[i] laten aangeven in welke set element ii zich bevindt. Hoe implementeer je dan de MAKESET, FINDSET en UNION operatie? Bespreek hun complexiteit en geef een zo klein mogelijke bovengrens voor de (asymptotische) uitvoertijd van een reeks van m operaties, waaronder n makeset operaties. Geef ook een voorbeeld dat deze bovengrens gerealizeerd kan worden.
3. FIBONACCI HEAPS: Stel dat een knoop xx in de root list van een Fibonacci heap 7 kinderen heeft, uit hoeveel knopen bestaat de boom waarvan x de root is dan minstens? Wat is de minimale diepte van deze boom?
4. FLOW NETWORKS: Stel dat alle edges een gehele capaciteit c(u,v)∈1,2,...,kc(u,v)∈1,2,...,k hebben. Geef dan een uitdrukking voor de asymptotische complexiteit van het preflow- push algoritme in termen van |V|, |E| and k (Hint: kijk vooral naar het aantal nonsaturating push operaties).
5. MINIMAL SPANNING TREES: Stel dat we volgende recursieve methode hanteren voor het bepalen van een minimale spanning tree TT van G=(V,E)G=(V,E). Splits VV in twee disjuncte sets V1V1 en V2V2 en bepaal de minimale spanning trees T1T1 en T2T2 van de twee grafen G1=(V1,E1)G1=(V1,E1) en G2=(V2,E2)G2=(V2,E2) met Ei=(u,v)|u,v∈ViEi=(u,v)|u,v∈Vi en definieer TT als de unie van T1T1, T2T2 en een edge met minimaal gewicht tussen V1V1 en V2V2. Bewijs de correctheid van dit algoritme of geef een tegenvoorbeeld.
6. BIPARTITE GRAPH MATCHING: Gegeven een bipartite graaf met |L|=|R||L|=|R| die een perfecte matching bevat. Hoeveel iteraties zal het algoritme van Ford en Fulkerson nodig hebben om een perfecte matching te vinden?

7. DARTS CHAMPIONSHIP: Stel dat we n darts spelers hebben en dat speler i reeds wi wedstrijden gewonnen heeft, voor i=l,...,ni=l,...,n. Stel verder dat er nog k wedstrijden te spelen zijn, waarbij elke wedstrijd tussen twee spelers gaat en wordt gewonnen door één van beide. Geef dan een snel algoritme om na te gaan of een speler (bv. de j-de) nog kampioen kan worden (d.i., minstens evenveel heeft gewonnen als gelijk welke andere speler nadat de k resterende wedstrijden gespeeld zijn).

Academiejaar 2009 - 2010 - 2de zittijd

1. Toon aan dat bij een topological sort van een graph met gesloten paden, het aantal foute edges niet altijd minimaal is. (Hint: een graph met 4 nodes was voldoende)
2. Toon aan dat c(u,v)+c(v,u)=fc(u,v)+fc(v,u)c(u,v)+c(v,u)=fc(u,v)+fc(v,u) met cc de capacity en fc de flow van een flow-netwerk.
3. Schrijf een algortime met complixiteit O(E∗logE)O(E∗logE) , dat in een flownetwork het augmenting path vind met de grootste capaciteit. (Hint: Sorteer edges op capacity en zoek met DFS ofdat E1,...,EKE1,...,EK een path vormt met E1 tot EK edges).
4. Gegeven een bipartite graph. Doe het Ford-Fulkerson algoritme en bepaal de kleinste bovengrens van het langste augmenting path dat je kan hebben.
5. Je hebt een niet geconnecteerde graph GG, met kk delen. Zoek een algoritme dat deze delen vindt en ze kan voorstellen (Hint : Disjoint Sets.)
6. Gegeven een spannings tree TT van graph GG. Amax(T)Amax(T) = de maximum weight van een edge. Bewijs dat als Amax(T)Amax(T) miminaal beschouwt wordt TT ook een minimal spannings tree is of geef een tegenvoorbeeld.
7. Gegeven een Fibonacci heap met 30 nodes. Wat is het maximale aantal root nodes die je kan overhouden nadat je een Delete-Min hebt uitgevoerd.

Academiejaar 2008 - 2009 - 1ste zittijd

Er is de keuze tussen een openboek- en geslotenboekexamen. Het openboekexamen heeft enkel de zeven oefeningen (met puntenverdeling 3, 3, 3, 2.5, 2.5, 3, 3) en het geslotenboekexamen heeft de drie theorievragen en de vier laatste oefeningen (met verdeling 3, 3, 4 voor de theorie en 2, 2, 3, 3) voor de praktijk).

Theorie

1. {Graph searching:}Wat verstaan we onder een topological sort van een graaf G=(V,E)G=(V,E), op welke klasse van grafen kan men deze uitvoeren en hoe kunnen we nagaan of een graaf tot deze klasse behoort? Hoe kunnen we een topological sort eenvoudig uitvoeren met behulp van een graph searching algoritme en bewijs de correctheid hiervan.
2. {Bipartite graph matching:}Toon aan dat wanneer ll de lengte van het kortste augmenting path is met betrekking tot een matching MM en P1∪…∪PkP1∪…∪Pk is een maximale knoopdisjuncte set van augmenting paden van lengte ll met betrekking tot MM, dan zal P′P′, een kortste augmenting pad met betrekking tot M′=M⊕(P1∪…∪Pk)M′=M⊕(P1∪…∪Pk) een lengte hebben van minstens l+1l+1.
3. {Fibonacci heap:}Leg uit hoe de {DeleteMin}operatie (en bijhorende {CleanUp}) werken. Bepaal ook de geamortiseerde kost van deze operatie.

Praktijk

1. {Graph searching:}Laat GG een ongerichte samenhangende graaf zijn zonder gesloten paden. Geef een O(|V|)O(|V|) algoritme om de diameter van GG te bepalen, deze is gedefinieerd als de lengte van het langst mogelijke pad tussen twee knopen.

2. {Disjoint-sets forest:}Stel dat we een array {PartOf}van nn elementen gebruiken en {PartOf$[i]$}laten aangeven in welke set element ii zich bevindt. Hoe implementeer je dan de {MakeSet}, {FindSet}en {Union}operatie? Bespreek hun complexiteit en geef een zo klein mogelijke bovengrens voor de (asymptotische) uitvoertijd van een reeks van mm operaties, waaronder nn {MakeSet}operaties. Geef ook een voorbeeld dat deze bovengrens gerealiseerd kan worden.

3. {Fibonacci heaps:}Stel dat we als potentiaalfunctie 2t(H)+2m(H)2t(H)+2m(H) hanteren in plaats van t(H)+2m(H)t(H)+2m(H), wat zou er dan veranderen in onze analyse van de geamortiseerde kost van de {DecreaseKey}en {DeleteMin}operatie?

4. {Flow networks:}Stel dat we aan elke knoop v∈Vv∈V eveneens een capaciteit c(v)≥0c(v)≥0 toekennen en als extra eis opleggen dat de hoeveelheid flow die in een knoop vv toekomt beperkt is door c(v)c(v). Kan je dan nog steeds de maximale flow op een efficiënte wijze bepalen? Geef voldoende argumenten om je antwoord te funderen.

5. {Minimal spanning trees:}{}Aanschouw volgend algoritme:

    1. sorteer de edges zodat w(e1)≥w(e2)≥…≥w(em)w(e1)≥w(e2)≥…≥w(em) met m=|E|m=|E| en stel T=GT=G;

    2. voor i=1i=1 tot mm: verwijder edge eiei uit TT indien deze TT niet onsamenhangend maakt.

    Levert dit een minimale spanning tree TT op? Geef een bewijs dat dit het geval is of geef een tegenvoorbeeld.

6. {Coloring images:}Stel dat we een grid bestaande uit n×nn×n witte pixels opgegeven hebben met tussen de pixels in enkele lijnen die de n2n2 pixels opdelen in enkele gebieden (zie bord). Van elke pixel pp weten we of er een lijn is tussen pp en zijn boven-, onder-, linker- of rechterbuur. Geef een snel algoritme dat de pixels inkleurt zodat alle gebieden een verschillende kleur hebben. Analyseer de complexiteit van je oplossing en bespreek je gebruikte datastructuur.

   > > (80,80)(0,0) (30,20)(10,0){4}{(10,10){[gray]{0.70}{\makebox(5,5){}}}} (30,30)(10,0){3}{(10,10){[gray]{0.70}{\makebox(5,5){}}}} (50,40){(10,10){[gray]{0.70}{\makebox(5,5){}}}} (40,60)(10,0){3}{(10,10){[gray]{0.90}{\makebox(5,5){}}}} (50,50)(10,0){2}{(10,10){[gray]{0.90}{\makebox(5,5){}}}} (10,50)(0,10){2}{(10,10){[gray]{0.30}{\makebox(5,5){}}}}
   > >
   > > {.3pt}(0,0)(0,10){9}{(1,0){80}} (0,0)(10,0){9}{(0,1){80}}
   > >
   > > {2pt}(30,20){(1,0){40}} (30,20){(0,1){20}} (30,40){(1,0){20}} (60,30){(0,1){20}} (50,40){(0,1){10}} (50,50){(1,0){10}} (70,20){(0,1){10}} (60,30){(1,0){10}}
   > >
   > > (40,70){(1,0){30}} (40,60){(1,0){10}} (40,60){(0,1){10}} (50,50){(0,1){10}} (60,50){(1,0){10}} (70,50){(0,1){20}}
   > >
   > > (10,70){(1,0){10}} (10,50){(1,0){10}} (10,50){(0,1){20}} (20,50){(0,1){20}}

7. {Assigning objects:}Stel dat we nn objecten hebben en pp personen die elk een eigen verlanglijstje hebben dat bestaat uit minstens één object. Geef een snel algoritme om na te gaan of het mogelijk is deze nn objecten te verdelen onder de pp personen zodat niemand over alle gewenste objecten beschikt. Wat is de complexiteit van je oplossing?