****

**Kuhn-Munkres algoritam**

**(Mađarski algoritam)**

**SEMINARSKI RAD**

**Predmet: Algoritmi i strukture podataka**

**Profesor: Doc. dr. Nermin Goran**

**Studenti: Husein Čišić, Belma Đelilović,**

**Nadina Kolić i Faris Trtak**

**Indeksi: 226 , 184 , 167 , 196**

**Studij: Softversko inžinjerstvo**

**Zenica, januar 2022**

**Sadržaj:**

**1.**[**UVOD**](#_40k7sp1rj2iq)………………………………………………………………………………………………………..2

[**2. GLAVNI DIO**](#_bkdq43j8bujl)……………………………………………………………………………………………………..3

[2.1 Razvoj](#_2fn0yxc9t3o0)………………………………………………………………………………………………… 3

[2.2 Algoritam preko matrice (minimum)](#_vgpd2noqsicw).......................................................... 4

[2.3 Algoritam preko matrice (maksimum)](#_vneyqbs6nqmu)........................................................ 7

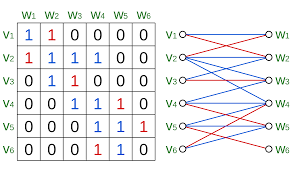
[2.4. Algoritam preko bipartitnog grafa](#_dkmbj8mkj2r7)………………………………………………………….. 9

[**3. ZAKLJUČAK**](#_br5d4daey5ah)……………………………………………………………………………..………………13

[**4. LITERATURA**](#_tn9i593lhybx)………………………………………………………………………………………………..….14

# UVOD

Mađarski algoritam ili algoritam dodjele je kombinatorno optimizacijski algoritam. Algoritam rješava problem dodjele u polinomskom vremenu i ovaj algoritam je anticipirao kasnije primalno-dualne metode. Harold Kuhn je razvio i objavio ovu metodu i ona je postala početna tačka brzog razvoja oblasti efikasne kombinatorne optimizacije. Pošto je mađarski algoritam brz i jednostavan, on je pogodan za sve probleme koji se mogu opisati kao “cjelobrojno programiranje”. Mađarski algoritam je široko rasprostranjen i koristi se u raznim problemima dodjele i uparivanja, kao što je smanjenje registra tokom alokacije složenih funkcionalnih jedinica, DNK računarstva, itd. Mađarski algoritam je algoritam prvobitne složenosti O(n4), ali se njegova složenost uspjela svesti na O(n3). Algoritam se može predstaviti na dva načina, pomoću matrice i bipartitnog grafa.



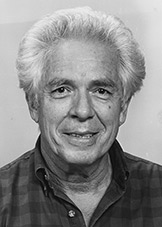
Slika 1. Kuhn-Munkres algoritam [1]

# 2. GLAVNI DIO

## 2.1 Razvoj

Razvio ga je i objavio 1955. Američki matematičar Harold W. Kuhn, koji mu je dao naziv "mađarski algoritam" jer je algoritam bio u velikoj mjeri zasnovan na ranijim radovima dvojice mađarskih matematičara Denesa Koniga i Jen ̋ o Egerváryja koji su bili izvor njegove inspiracije.

1957. godine američki matematičar James R. Munkres je počeo proučavati algoritam i primijetio da je polinomski. Od tada je algoritam poznat i kao Kuhn–Munkres algoritam ili Munkresov algoritam dodjele. Vremenska složenost originalnog algoritma bila je O(n4), međutim Američki informatičari Jack Edmonds i Richard Karp, te Tomizawa, primijetili su da se može modificirati kako bi se postigla O(n3) složenost. Jedna od najpopularnijih O(n3) varijanti jeste Jonker–Volgenant algoritam. Lester R. Ford i Delbert R. Fulkerson proširili su metodu na opšte probleme maksimalnog protoka u obliku Ford-Fulkerson algoritma. Ali 2006. godine otkriveno je da je njemački matematičar Carl Gustav Jacob Jacobi zapravo riješio ovaj problem u 19. vijeku, tačnije 1840. godine, a rješenje je objavljeno nakon njegove smrti 1890. godine na latinskom jeziku.

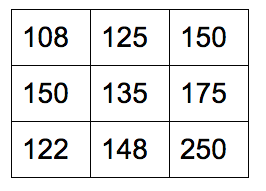


Slika 2. i 3. Harold W. Kuhn i James R. Munkres [2][3]

## 2.2 Algoritam preko matrice (minimum)

Korak 1:

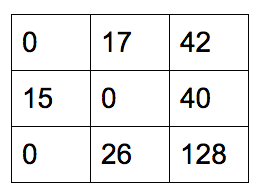
Prvo se provjerava da li je matrica kvadratna tj. n\*n. Ako nije dodaje se lažni red/kolona. Ako je matrica kvadratna onda se prelazi na korak 2.



Slika 4. Korak 1 [4]

Korak 2:

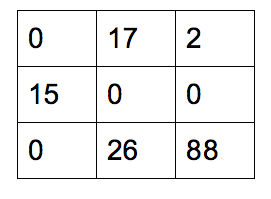
Od svake vrijednosti u redu se oduzima minimalna vrijednost tog reda. Ovo se uradi za svaki red.



Slika 5. Korak 2 [5]

Korak 3:

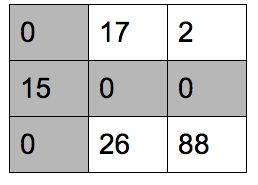
Od svake vrijednosti u koloni se oduzima minimalna vrijednost te kolone. Ovo se uradi za svaku kolonu. Ako je minimalna vrijednost u koloni 0 onda se ta kolona preskaće.



Slika 6. Korak 3 [6]

Korak 4:

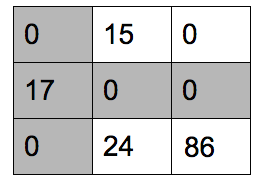
Preko nula se povlači najmanji mogući broj linija. Ako je broj linija jednak vrijednosti n, odmah se može preći na zadnji korak. Ako nije, ide se na sljedeći korak.



Slika 7. Korak 4 [7]

Korak 5:

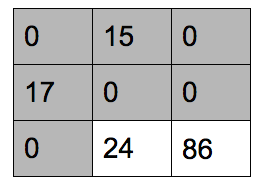
Od svih vrijednosti u matrici koje nisu prekrivene linijama nalazi se najmanja vrijednost. Ta najmanja vrijednost se oduzima od vrijednosti koje nisu prekrivene linijama, a dodaje se onim vrijednostima na kojima je presjek dvije linije. Vrijednosti preko kojih prelazi jedna linija ostaju iste.



Slika 8. Korak 5 [8]

Korak 6:

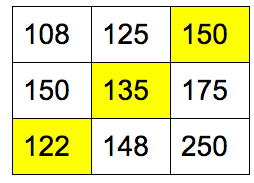
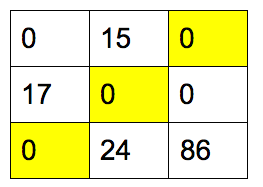
Opet se preko nula povlači najmanji broj linija. Ako je broj linija jednak vrijednosti n, onda ima određeno rješenje i algoritam staje. Ako broj linija nije jednak n, vraća se na korak 5.



Slika 9. Korak 6 [9]

Korak 7:

U ovom koraku se posmatraju nule i određuju se najmanje moguće vrijednosti.



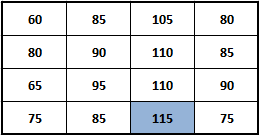
Slika 10. i 11. Korak 7 [10][11]

## 

## 2.3 Algoritam preko matrice (maksimum)

Korak 1:

Pronaći najveći broj u “početnoj” matrici.

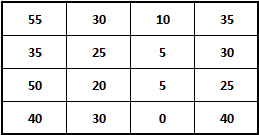


Slika 12. Korak 1 [12]

Korak 2:

Od najvećeg broja iz prethodnog koraka treba oduzeti svaku vrijednost iz svake kolone.

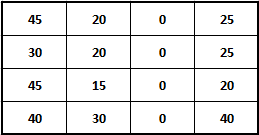
(da bi se početna matrica pretvorila u minimum matricu).



Slika 13. Korak 2 [13]

Korak 3:

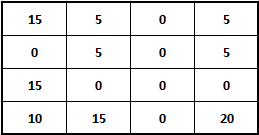
Od svake vrijednosti u redu se oduzima minimalna vrijednost tog reda. Ovo se uradi za svaki red.



Slika 14. Korak 3 [14]

Korak 4:

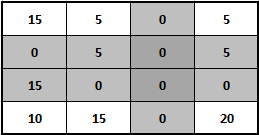
Od svake vrijednosti u koloni se oduzima minimalna vrijednost te kolone. Ovo se uradi za svaku kolonu. Ako je minimalna vrijednost u koloni 0 onda se ta kolona preskaće.



Slika 15. Korak 4 [15]

Korak 5:

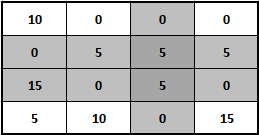
Preko nula se povlači najmanji mogući broj linija. Ako je broj linija jednak vrijednosti n, odmah se može preći na zadnji korak. Ako nije, ide se na sljedeći korak.



Slika 16. Korak 5 [16]

Korak 6:

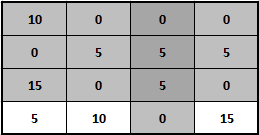
Od svih vrijednosti u matrici koje nisu prekrivene linijama nalazi se najmanja vrijednost. Ta najmanja vrijednost se oduzima od vrijednosti koje nisu prekrivene linijama, a dodaje se onim vrijednostima na kojima je presjek dvije linije. Vrijednosti preko kojih prelazi jedna linija ostaju iste.



Slika 17. Korak 6 [17]

Korak 7:

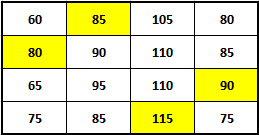
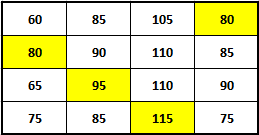
Opet se preko nula povlači najmanji broj linija. Ako je broj linija jednak vrijednosti n, onda ima određeno rješenje i algoritam staje. Ako broj linija nije jednak n, vraća se na korak 6.



Slika 18. Korak 7 [18]

Korak 8:

U ovom koraku se posmatraju nule i određuju se najveće moguće vrijednosti pomoću matrice. Prije toga matricu treba napisati u obliku početne matrice da bi se mogle pronaći maksimalne vrijednosti.



Slika 19. i 20.. Korak 8 [19][20]

## 2.4. Algoritam preko bipartitnog grafa

Ulaz: bipartitni graf G sa biparticijom (X,Y) , |X| = |Y|.

Izlaz: savršeno sparivanje M u G ili skup S ⊆ X za koji je |N(S) < |S|.

Korak 1:

Neka je M bilo koje sparivanje od G, npr M = ∅.

Korak 2:

Ako je X M-zasićen, stop (M je tada savršeno sparivanje od G). U suprotnom, neka je vrh x ∈ X M-nezasićen. Stavimo S = {x}, T = ∅.

Korak 3:

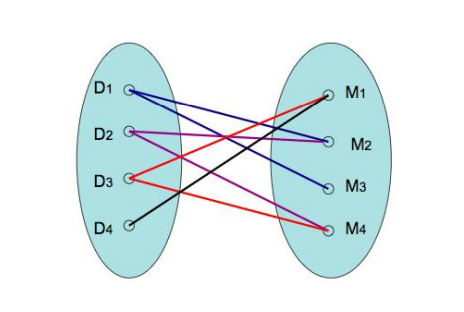
Ako je N(S) = T, stop jer je |N(S)| < |S|. U suprotnom, neka je y ∈ N(S) \ T.

Korak 4:

Ako je y ∈ Y M-zasićen vrh, neka je zy ∈ M. Zamijeniti S sa S ∪ {z}, a T sa T ∪ {y} i vratiti se na korak 3. Ako je y M-nezasićen vrh, neka je P M-uvećavajući (x,y)-put. Zamijeniti M sa M △ E(P) i vratiti se na korak 2.

Primjer:

U nekom društvu, koje se sastoji od četiri djevojke D1, D2, D3, D4 i četiri momka M1, M2, M3, M4 postoje neke uzajamne simpatije. Naime, djevojka D1 simpatizira momke M1 i M3, djevojka D2 momke M2 i M4, djevojka D3 momke M1 i M4, te djevojka D4 simpatizira momka M1. Odrediti koliko najviše parova se možeme povezati poštujući navedene simpatije?



Slika 21. Bipartitan graf - postavka zadatka [21]

Rješenje:

Ulaz: bipartitni graf G(X,Y), X = {D1, D2, D3, D4}, Y = {M1, M2, M3, M4}.

Korak 1:

Neka je M = ∅.

Korak 2:

X nije M- zasićen. Vrh x = D1 ∈ X je M- nezasićen. Stavljamo da je S = {D1}, T = ∅.

Korak 3:

N(S) = {M2, M3} ≠ T = ∅. Neka je y = M2 ∈ N(S) \ T = {M2, M3}.

Korak 4:

y je M-nezasićen vrh. Neka je P = D1M2 M-uvećavajući put. Dakle M = {D1M2} i vratimo se na korak 2.

Korak 2:

X je M = {D1M2}-nezasićen. Vrh x = D2 ∈ X je M-nezasićen. Stavljamo da je S = {D2}, T = ∅.

Korak 3:

N(S) = {M3, M4} ≠ T = ∅. Neka je y = M2 ∈ N(S) \ T = {M2, M4}.

Korak 4:

y = M2 je M-zasićen vrh jer je D1M2 ∈ M. S = S ∪ {D1} = {D1, D2}, T = T ∪ {M2} = {M2}. Vratimo se na korak 3.

Korak 3:

N(S) = N({D1, D2}) = {M2, M3, M4} ≠ T = {M2}. Neka je y = M3 ∈ N(S) \ T = {M3, M4}.

Korak 4:

y = M3 je M-nezasićen. Neka je P = D2M2D1M3 M-uvećavajući (D2, M3)-put.

M = M △ E(P) = {D1M3, D2M2} i vratimo se na korak 2.

Korak 2:

X je M ={D1M3, D2M2}-nezasićen. Vrh x = D3 ∈ X je M-nezasićen. Stavimo da je S = {D3}, T = ∅.

Korak 3:

N(S) = {M1, M4} ≠ T = ∅. Neka je y = M1 ∈ N(S) \ T = {M1, M4}.

Korak 4:

y = M1 je M-nezasićen vrh. P = D3M1 je M-uvećevajući (D3,M1)-put, E(P) = {D3M1}

M = M △ E(P) = {D1M3, D2M2, D3M1} i vratimo se na korak 2.

Korak 2:

X je M ={D1M3, D2M2, D3M1}-nezasićen. Vrh x = D4 ∈ X je M-nezasićen. Stavimo da je S = {D4}, T = ∅.

Korak 3:

N(S) = {M1} ≠ T = ∅. Neka je M1 ∈ N(S) \ T = {M1}

Korak 4:

y = M1 je M-zasićen vrh jer je D3M1 ∈ M. S = S ∪ {D3} = {D3, D4}, T = T ∪ {M1} = {M1}. Vratimo se na korak 3.

Korak 3:

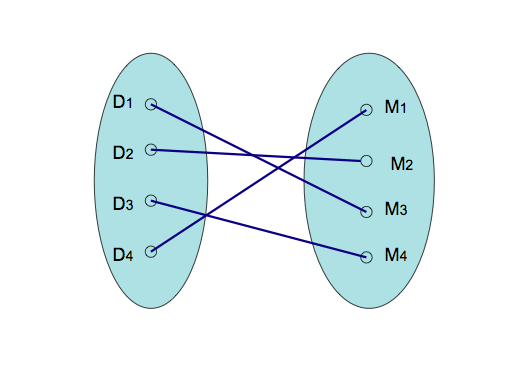
N(S) = N({D3, D4}) = {M1, M4} ≠ T = {M1}. Neka je y = M4 ∈ N(S) \ T = {M4}.

Korak 4:

y = M4 je M-nezasićen. P= D4M1D3M4 je M-uvećavajući (D4M5)-put, E(P) = {D4M1, D3M1, D3M4}, M = M △ E(P) = {D1M3, D2M2, D3M4, D4M1}, vratimo se na korak 2.

Korak 2:

X je M = {D1M3, D2M2, D3M4, D4M1}-zasićen. Dakle M je savršeno sparivanje.



Slika 22. Bipartitan graf - rješenje [22]

# 3. ZAKLJUČAK

Mađarski algoritam tj. Algoritam dodjele se većinom koristi u svrhu dodjele posla koristeći podudaranje “jedan za jedan” gdje se svaki posao mora dodijeliti samo jednoj mašini, kako bi se pronašla najjeftinija ili najskuplja cijena. Ali ovaj algoritam se može koristi u različite svrhe kao što je računanje cijene, vremena, dužine pređenog puta, itd. Ustvari, algoritam dodjele se može koristiti u bilo kojem scenariju gdje se treba odrediti minimalna ili maksimalna vrijednost neke veličine. Algoritam dodjele je O(n4)/O(n3) koji se može predstaviti pomoću kvadratne matrice ili bipartitnog grafa.

# 

# 4. LITERATURA

<http://e.math.hr/math_e_article/br17/kalebic_et_al>

<https://brilliant.org/wiki/hungarian-matching/>

<https://hr.economy-wiki.com/11033922-hungarian-method>

<https://hrcak.srce.hr/file/100534>

<https://bs.warbletoncouncil.org/metodo-hungaro-5228>

<https://www.twblogs.net/a/5b800bf92b717767c6b304e3?fbclid=IwAR2oFf1Mv9PI3oiCoAHWHmS9qxfgZIFqdIA4fZRTGT9j9v7w-NP4kHedKVQ>

[https://journals.sagepub.com/doi/pdf/](https://journals.sagepub.com/doi/pdf/10.1260/1748-3018.8.2.219)

Slike:

<https://ds055uzetaobb.cloudfront.net/brioche/uploads/LviKNwDdBm-matching.png?width=1200>

<https://upload.wikimedia.org/wikipedia/en/b/b4/Harold_W._Kuhn.jpg>

<https://i.pinimg.com/originals/22/64/49/22644956e605a6f94dfeea6e5b722262.jpg>

<https://ds055uzetaobb.cloudfront.net/brioche/uploads/W1mXMcmOzN-screen-shot-2016-06-26-at-24958-pm.png?width=1200>

<https://ds055uzetaobb.cloudfront.net/brioche/uploads/oKw5rsJSy7-screen-shot-2016-06-26-at-25138-pm.png?width=1200>

<https://ds055uzetaobb.cloudfront.net/brioche/uploads/NX2rDsLoXU-one.png?width=1200>

<https://ds055uzetaobb.cloudfront.net/brioche/uploads/AgpL08MXBW-two.png?width=1200>

<https://ds055uzetaobb.cloudfront.net/brioche/uploads/Qo1Qrlyqij-four.png?width=1200>

<https://ds055uzetaobb.cloudfront.net/brioche/uploads/4dWcc8zNPF-five.png?width=1200>

<https://ds055uzetaobb.cloudfront.net/brioche/uploads/SQXyyAvfcO-six.png?width=1200>

<https://ds055uzetaobb.cloudfront.net/brioche/uploads/D3Clbwya49-screen-shot-2016-06-26-at-30740-pm.png?width=1200>

<https://ds055uzetaobb.cloudfront.net/brioche/uploads/AgpL08MXBW-two.png?width=1200>