

Nonogram

Anja Svečarovski, JMBAG 0145027237 (odgovorna osoba)

Lana Kohut, JMBAG 0009069158

Daniel Katić, JMBAG 0303123347

10. veljače 2025.

Sadržaj

1	Uvod	3
1.1	Sažetak teme	3
1.2	Ciljevi projekta	3
1.3	Očekivani ishod	4
2	Strategije rješavanja	4
2.1	Definiranje varijabli	4
2.2	Potpuna popuna	5
2.3	Segmentacija	6
2.4	Identifikacija zajamčenih ispunjenja	8
2.5	Označavanje praznih kvadrata	10
2.6	Provjera granica	11
2.7	Kombiniranje informacija	12
3	Zaključak	14
4	Prilozi	15
5	Literatura	16

1 Uvod

Nonogrami, također poznati kao Hanjie, Paint by Numbers, Picross, Griddlers i Pic-a-Pix, te pod raznim drugim imenima, slikovne su logičke zagonetke u kojima ćelije u mreži moraju biti obojene ili ostavljene praznima prema brojevima sa strane kako bi se otkrila skrivena slika nalik pikselnoj umjetnosti. U ovoj vrsti slagalice, brojevi su oblik diskretne tomografije koja mjeri koliko neprekinutih linija popunjenih kvadrata ima u bilo kojem retku ili stupcu. Na primjer, trag "4 8 3" bi značio da postoje skupovi od četiri, osam i tri ispunjena kvadrata, tim redoslijedom, s najmanje jednim praznim kvadratom između uzastopnih skupova.

Nonogrami nisu samo zabavni, već i izvrstan alat za razvijanje kritičkog razmišljanja, što ih čini idealnim kandidatom za analizu u obrazovnim kontekstima[MM].

1.1 Sažetak teme

Nonogrami su logički problemi u kojima se koriste brojevi kako bi se rješavao uzorak unutar mreže. Brojevi prikazani s vanjske strane pokazuju koliko je ćelija ispunjenih u tom retku/stupcu. Više brojeva označava da postoji barem jedan prazan kvadrat između.

Brojevi u Nonogramima mogu se smatrati ograničenjima koja oblikuju prostor rješenja, što je koncept koji Michalewicz analizira kao temeljni dio logičkih zagonetki[MM].

1.2 Ciljevi projekta

Cilj ovog projekta je:

- Razviti algoritam za rješavanje Nonograma koristeći propozicijsku logiku i teoriju skupova.
- Vizualizirati proces rješavanja i prikazati konačna rješenja.
- Analizirati dobivene rezultate s obzirom na efikasnost i točnost algoritma.

Teorija skupova omogućuje formalno modeliranje mogućih rješenja logičkih zagonetki, što je osnova algoritamskog pristupa koji se koristi u ovom projektu. Rosen objašnjava kako se koncepti diskretne matematike, poput skupova i relacija, mogu koristiti za rješavanje problema optimizacije i logike, što je u središtu ovog projekta[Ros].

1.3 Očekivani ishod

Rezultat projekta bit će funkcionalni algoritam za rješavanje Nonograma, detaljna analiza rješenja te dokumentacija metodologije korištene u razvoju.

2 Strategije rješavanja

U ovoj sekciji opisujemo metodološki pristup korišten u razvoju i analizi Nonograma. Cilj je predstaviti korake koje smo poduzeli kako bismo dizajnirali, riješili i evaluirali Nonograme, uz detaljan pregled tehnika korištenih u svakoj fazi procesa.

Michalewicz opisuje kako sistematizacija rješavanja problema, poput razbijanja složenih zadataka na jednostavnije korake, može značajno poboljšati razumijevanje i rješavanje zagonetki [MM].

2.1 Definiranje varijabli

Nonogram se sastoji od mreže kvadrata, gdje svaki kvadrat može biti popunjen ili prazan. U našem modelu koristimo logičke varijable kako bismo prikazali stanje pojedinih kvadrata:

Logička vrijednost	Značenje	Prikaz
\top (True)	Kvadrat je popunjen	■
\perp (False)	Kvadrat je prazan	.

Tablica 1: Prikaz logičkih vrijednosti u modelu Nonograma

Svaki kvadrat u mreži modeliramo varijablom x_L , gdje:

- x_L označava kvadrat u retku ili stupcu duljine L
- $x_L = \top$ ako je kvadrat popunjen
- $x_L = \perp$ ako je kvadrat prazan

Ovo možemo formalizirati kao:

$$x_L = \begin{cases} \top, & \text{ako je kvadrat na poziciji } (L) \text{ popunjen} \\ \perp, & \text{ako je kvadrat na poziciji } (L) \text{ prazan} \end{cases}$$

Svaki redak i stupac imaju pridružene brojeve (tragove) koji definiraju uzastopne sekvence popunjenih kvadrata.

- n_1, n_2, n_3 su brojevi koji označavaju duljine uzastopnih popunjenih kvadrata
- Svaka sekvenca popunjenih kvadrata mora biti odvojena barem jednim praznim kvadratom

Budući da smo ograničili mrežu na dimenzije 5×5 , u svakom retku i stupcu može postojati najviše 3 traga (n_1, n_2, n_3) , jer veće sekvence ne bi stale unutar pet kvadrata.

Prema pravilima Nonograma, između svake dvije popunjene sekvence mora postojati barem jedno prazno polje (r). To znači da ako imamo dvije sekvence popunjenih kvadrata n_1 i n_2 , tada između njih mora postojati barem jedno prazno polje (r):



Uvodimo također i skupove S_i , gdje svaki skup S_i predstavlja skup kvadrata koji su popunjeni kada se sekvenca smjesti na i -tom mogućem položaju. Budući da postoji više mogućih položaja za istu sekvencu, skupovi S_1, \dots, S_m predstavljaju sve moguće načine smještanja sekvence unutar retka ili stupca.

Ova pravila omogućuju precizno definiranje uvjeta za rješavanje Nonograma i postavljanje strategija za njegovo rješavanje koje ćemo definirati u nastavku.

2.2 Potpuna popuna

Kada broj uz redak ili stupac odgovara duljini retka/stupca, svi kvadrati se popunjavaju.

Što također možemo napisati da kada broj ispunjenih kvadrata n u opisu retka ili stupca odgovara duljini L tog retka ili stupca, svi kvadrati u njemu moraju biti popunjeni.

Ako su kvadrati u retku označeni s x_1, x_2, \dots, x_L , tada vrijedi:

$$x_1 \wedge x_2 \wedge \dots \wedge x_L = \top$$

Koraci za primjenu:

1. Provjeri svaki redak i stupac u mreži.
2. Ako je $n = L$ onda označi sve kvadrate u tom retku ili stupcu kao popunjene (\top).
3. Nastavi s drugim redovima/stupcima dok ne prođeš cijelu mrežu.

Primjer: Veličina mreže nam je 5. I ako imamo $n = 5$ to znači da nam sva polja moraju biti spremljena kao \top .



Slika 1: Strategija Potpuna popuna

2.3 Segmentacija

Ostavlja se prazan prostor između sekvenci kako bi se osigurala pravilna raspodjela.

Duljina retka/stupca L se uspoređuje sa duljinom sekvence n i praznim poljem između sekvenci. Između sekvenci mora postojati barem jedan prazan kvadrat.

Ukupna duljina opisa: $n_1 + r + n_2 + r + n_3$ se uspoređuje s ukupnom duljinom retka/stupca L i ukoliko su jednake, popunjavaju se puna i prazna polja.


Koraci za primjenu:

1. Analiziraj opis i identificiraj duljine sekvenci $n_1 \dots n_3$
2. Osiguraj minimalne razmake između sekvenci:
 - (a) Dodaj $r = 1$ prazan kvadrat između svake sekvence.
3. Popuni kvadrate prema minimalnim zahtjevima opisa.

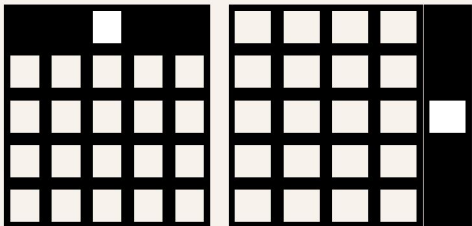
Primjer: Veličina mreže nam je 5. I ako imamo 2,2 to znači da imamo

$$\begin{array}{c} 2 + 1 + 2 \\ (n_1 + r + n_2) \end{array}$$

Što znači da ovom provjerom imamo popunjen cijeli red/stupac sa jednim praznim mjestom između njih. Puna polja označavamo sa \top a prazno polje sa \perp (tako da znamo da je to provjereno i sigurno prazno polje).

02 Segmentacija 

Ako imamo mrežu 5x5 i $n=2,2$ iznad nekog retka ili stupca, onda trebamo označiti 2 kvadrata za redom kao popunjena, odnosno True, pustiti jedan prazan i označiti druga dva kao popunjena, odnosno True.



Slika 2: Strategija Segmentacija

2.4 Identifikacija zajamčenih ispunjenja

Svaka sekvenca duljine n mora zauzeti n uzastopnih kvadrata. Budući da je veličina retka (ili stupca) L , početna pozicija sekvence može varirati od 1 do $L - n + 1$. To znači da postoji više mogućih načina smještanja sekvence unutar retka ili stupca, uzimajući u obzir da se kvadrati popunjavaju uzastopno ovisno o tragu n i da moramo imati 1 slobodno mjesto ako imamo dva traga.

Uvodimo skupove S_i , gdje svaki skup S_i predstavlja skup kvadrata koji su popunjeni kada se sekvenca smjesti na i -tom mogućem položaju. To nam označava različite kombinacije položaja tragova.

Ukoliko ukupna duljina sekvence zajedno sa zajamčenim praznim mjestom nije ista duljini retka ili stupca, postoji više mogućih položaja za istu sekvencu, skupovi S_1, \dots, S_m predstavljaju sve moguće načine smještanja sekvence unutar retka ili stupca.

Ako postoji više mogućih položaja sekvence, kvadrati koji se nalaze u preklapanju svih mogućih položaja su zajamčeno popunjeni. Taj zajamčeno popunjeni skup kvadrata P dobivamo kao presjek svih kombinacija S_i :

$$P = \bigcap_{i=1}^m S_i$$

gdje su S_i skupovi mogućih položaja sekvence, a P je skup zajamčeno popunjenih kvadrata.

Koraci za primjenu:

1. Identificiraj sve moguće položaje sekvence unutar retka ili stupca (izračun privremenih skupova S_1, \dots, S_m)
2. Pronađi preklapanje između svih mogućih položaja.
3. Označi preklapajuće kvadrate kao popunjene \top .

Primjer: Veličina mreže nam je 5. I ako imamo tragove $n = (2, 1)$ to znači da nam mogu biti sljedeća moguća rješenja (skupovi S_1, S_2, S_3):

$$S_1 = [\top, \textcircled{\top}, \perp, \top, \perp]$$

ili

$$S_2 = [\top, \textcircled{\top}, \perp, \perp, \top]$$

ili

$$S_3 = [\perp, \textcircled{\top}, \top, \perp, \top]$$

Vidimo da je druga pozicija uvijek popunjena u svim mogućim rješenjima (skupovi kombinacija), što znači da je zajamčeno popunjena, dok ostale pozicije ovise o specifičnoj instanci rješenja.

Gornje rješenje može se napraviti petljom, ali mi ćemo koristiti posebnu Python biblioteku `itertools.combinations(iterable, r)`. Generira sve moguće kombinacije duljine `r` od elemenata u `iterable`, bez ponavljanja elemenata. Python Itertools je biblioteka u Pythonu koja se sastoji od više metoda koje se koriste u različitim iteratorima za izračunavanje brzog i kodnog učinkovitog rješenja.

```
kombinacije = []
for raspodjela in itertools.combinations(range(
    duljina - trag[0] + 1), 1):
    mreza = [False] * duljina
    for i in raspodjela:
        for j in range(trag[0]):
            mreza[i + j] = True
    kombinacije.append(mreza)
return kombinacije
```

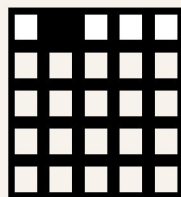
03

Identifikacija zajamčenih ispunjenja

Ako imamo mrežu 5×5 i $n=2,1$ iznad nekog retka ili stupca, onda znači da imamo više mogućnosti kako možemo ispuniti kvadrate:

```
[True, True, False, True, False]
ili
[True, True, False, False, True]
ili
[False, True, True, False, True]
```

Što znači da ovom provjerom nam je samo druga pozicija uvijek True (bez obzira na finalni točan odgovor) i spremamo X za to polje.



Slika 3: Strategija Identifikacije zajamčenih ispunjenja

2.5 Označavanje praznih kvadrata

S prijašnjim smo strategijama već popunili dijelove mreže ovisno o sekvenci (tragovima), međutim pošto provjeravamo posebno redove i stupce, može nam se desiti da imamo rješenje ali nemamo korak do njega.

Čovjek u glavi lakše primjenjuje sve ove strategije i zapravo ih koristimo u isto vrijeme, ali za definirati korake u Python kodu odlučili smo dodati korak označavanja praznih kvadrata kako bismo iskoristili informacije koje su nam dostupne u trenutnoj verziji mreže kako bismo označili sigurno prazne kvadrate.

Ako je sekvenca n već ispunjena unutar određenih granica, preostale pozicije izvan tih granica su prazne.

Koraci za primjenu:

1. Ako trag n zauzima maksimalnu moguću duljinu u retku/stupcu: označavamo sve kvadrate prije početka i nakon kraja sekvence kao \perp .
2. Ako tragovi $n = [n_1, n_2]$ zahtijevaju dvije sekvence ispunjenih kvadrata s razmakom između njih, prazni kvadrati se mogu odrediti prema udaljenosti između već ispunjenih dijelova:
 - (a) Ako su n_1 i n_2 već \top , sve ostale pozicije su \perp .
3. Ako trag n nije u potpunosti ispunjen, ali su poznati njegovi dijelovi, označavamo prazne kvadrate prije i poslije poznate sekvence, ako su ti kvadrati izvan granica mogućeg dosega.

Primjer: Pretpostavimo da imamo trag $n = 3$, a mreža duljine $L = 5$ izgleda ovako:

$$[\top, \top, \top, _, _]$$

Budući da je trag duljine 3 već ispunjen, svi preostali kvadrati moraju biti prazni:

$$[\top, \top, \top, \perp, \perp]$$

04 Označavanje praznih kvadrata

PRAVILO MAKSIMALNE SEKVENCE

- Trag $t = 3$, red $P = \{p1, p2, p3, p4, p5\}$, True su $p2, p3, p4$.
- Kvadrati $p1$ i $p5$ su prazni: $PF\ else = P \setminus \{p2, p3, p4\}$.

PRAVILO RAZMAKA

- Trag $T = [1, 1]$, red $P = \{p1, p2, p3, p4, p5\}$, True su $p2$ i $p4$.
- Kvadrati $p1, p3, p5$ su prazni jer nema mjesta za druge ispunjene kvadrate.

PRAVILO NEPOTPUNE SEKVENCE

- Trag $t = 3$, red $P = \{p1, p2, p3, p4, p5\}$, True je $p3$.
- Kvadrati $p1$ i $p2$ su prazni jer trag ne može započeti na tim pozicijama.

Slika 4: Strategija Označavanje praznih kvadrata

2.6 Provjera granica

Provjera granica odnosi se na označavanje kvadrata kao pune kada imamo već ispunjene neke kvadrate ali nisu uzastopni i označavanju praznih kada sekvenca ne može doseći određene kvadrate unutar retka ili stupca. Granice sekvence definiraju sve moguće pozicije koje sekvenca može zauzeti unutar mreže duljine L .

Koraci za primjenu:

1. Obrada redaka i stupca
 - (a) Prvo se pronalaze sve već popunjene (\top) i prazne (\perp) ćelije.
 - (b) Ako trag sadrži samo jedan segment, popunjavaju se sve ćelije unutar njegovih granica.
 - (c) Ako trag sadrži više segmenata, popunjene ćelije se ne spajaju, ali se prazne ćelije označavaju na granicama.
2. Prijenos informacija između redaka i stupaca
 - (a) Nakon što je red riješen - ako su svi njegovi tragovi potvrđeni, koristi se za ažuriranje pripadajućih stupaca.
 - (b) Popunjene ćelije u stupcima ažuriraju redove prema pravilima traga.

Primjer: Razmatramo sekvencu $n = 3$ unutar retka duljine $L = 5$. Ako su kvadrati x_2, x_4 već popunjeni (\top), onda se kvadrat x_3 također popunjava jer moramo imati 3 uzastopno popunjena kvadrata, a kvadrati p_1 i p_5 označeni kao prazni (\perp), jer se sekvenca ne može proširiti na te pozicije.



Slika 5: Strategija Provjere granica

2.7 Kombiniranje informacija

Ova strategija koristi dostupne informacije unutar retka ili stupca kako bi se završilo popunjavanje mreže. Ključni cilj je usporediti trenutno poznate popunjene kvadrate (\top) s tragovima koji definiraju raspored popunjenih polja te ažurirati nepoznata polja u skladu s pravilima igre. Ovo je zadnji korak koji "popunjava rupe" jer smo u prethodnim koracima definirali okvire.

Koraci za primjenu:

1. Za svaki redak:
 - Prebrojati već označene \top vrijednosti.
 - Ako odgovaraju zbroju tragova n , označiti preostala nepoznata polja kao prazna (\perp).
 - Ako su manji od ukupnog broja tragova, preostala nepoznata polja postupno označiti kao \top dok se ne dostigne potrebna količina.
2. Ponoviti isti postupak za svaki stupac.

Primjer Za mrežu veličine $L = 5$ s tragom $n = 2, 1$, pretpostavimo sljedeće trenutno stanje:

$[?, \top, \perp, \top, \perp]$

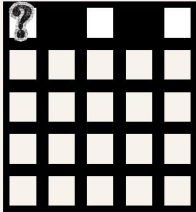
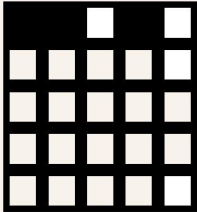
- Broj označenih \top vrijednosti je 2.
- Trag zahtijeva ukupno 3 popunjena polja \top .
- Polje sadrži 2 prazna polja \perp .
- Preostalo nepoznato polje može se označiti kao \top .

Ova pravila se iterativno primjenjuju kako bi se mreža postupno popunjavala na temelju poznatih informacija.

06

Kombiniranje informacija

Razmatramo mrežu veličine 5 i sekvencu 2, 1.
Stanje mreže u datom trenutku je
 $[?, \text{True}, \text{False}, \text{True}, \text{False}]$
Pregledom traga i sume True vrijednosti i
sigurnih praznih mjesta utvrđujemo da
preostalo mjesto možemo naznačiti kao
(True).

Slika 6: Strategija Kombiniranja informacija

3 Zaključak

Nonogrami nisu samo zabavne slagalice, već i primjer primjene logičkog zaključivanja i diskretne matematike u stvarnim problemima. Projekt pokazuje kako se matematički principi mogu intuitivno koristiti za rješavanje problema. Nadalje, istraživanje ovih metoda može se proširiti na druge logičke igre ili čak na područja kao što su optimizacija i računalna znanost. Kroz ovaj projekt, stječemo dublje razumijevanje logike i njezine praktične primjene, ističući važnost strukturiranog i analitičkog razmišljanja.

Ovaj projekt demonstrira kako se diskretna matematika, prema Rosenu, može koristiti za modeliranje i rješavanje problema koristeći formalne metode, čineći zagonetke poput Nonograma korisnim primjerima matematičkog pristupa problemima[Ros].

4 Prilozi

1. Infografika (vizualni prikaz strategija)
2. Python kod
3. Finalan poster

5 Literatura

- [MM] D.F. Michalewicz Matthew Michalewicz. *Puzzle-Based Learning: An Introduction to Critical Thinking, Mathematics, and Problem Solving.*
- [Ros] Kenneth H. Rosen. *Discrete Mathematics and Its Applications.*