RTK 2015 – Naloge in rešitve

Janez Brank

1.1 Delni izid

- Dano je zaporedje košev, npr.

$$-2, -2, +2, +2, +3, +2, -2, +2, -2, -3, +2, -2, -2, -2, +2,$$

- Najdi strnjeno podzaporedje z največjo vsoto (po absolutni vrednosti)
- Preprosta rešitev:

```
naj = 0;
for i = 1 to n:
    vsota = 0;
    for j = i to n:
        vsota = vsota + a[j];  // zdaj je to vsota a[i…j]
        if  vsota| > naj: naj = vsota;
```

1.1 Delni izid

- Boljša rešitev:
  - Najprej si pripravimo zaporedje delnih vsot s[i] = vsota a[1..i]
    s[0] = 0
    for i = 1 to n: s[i] = s[i – 1] + a[i]
  - Zdaj velja a[i] + a[i+1] + … + a[j] = s[j] – s[i – 1]
  - Če hočemo, da bo|s[j] – s[i – 1]| čim večja, moramo v zaporedju s[0..n] poiskati najmanjši in največji element

1.2 Kompresija

- Dobivamo zaporedje meritev, npr.

$$310, 310, 310, 310, 310, 311, 311, 311, 313, 313, 311.$$

- Sproti jih moramo sporočati naprej, vendar:
  - Od vsake skupine več zaporednih enakih meritev sporočimo le prvo, tretjo, peto, sedmo, …
- Rešitev:
  - Zapomniti si moramo dve stvari
    - Vrednost prejšnje meritve
    - Ali smo prejšnjo meritev sporočili naprej
  - Če smo prejšnjo sporočili naprej in ji je trenutna meritev enaka, potem trenutne ne sporočimo, sicer pa jo

1.3 znajdi.se

```
G S
- mimo O nadaljujemo skozi tri križišča, dokler ne pridemo do A
- pri B zavijemo desno, dokler ne dospemo do S
- iz A zavijemo v krožno križišče, od tam nadaljujemo naravnost do D
- na H zavijemo desno proti O
- od D sledimo modrim oznakam, dokler ne zagledamo B
- nato pri J peljemo naprej 1,2 km proti H
- iz G zavijemo levo proti J
```

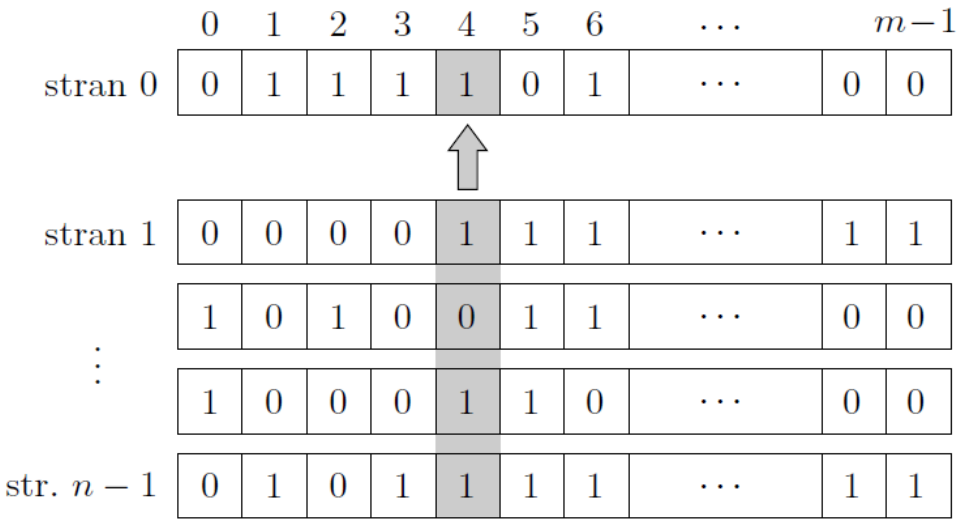
- Prebrati moramo opis poti, kjer so koraki premešani
  - Vsaka vrstica opisuje en korak poti
  - In vsebuje natanko dve veliki črki – prva je začetni, druga končni kraj na tem koraku
  - Rekonstruirati moramo pravi vrstni red krajev na poti
- Rešitev:
  - Pri branju ignoriramo vse znake, ki niso velike črke
  - Pripravimo si tabelo *naslednik*[26], v kateri bo za vsak kraj od A do Z pisalo, kateri je njegov naslednik na poti (na začetku postavimo vse na –1)
  - Ko naletimo na prvo veliko črko, si jo zapomnimo (recimo v spremenljivki *c1*)
  - Ko naletimo na drugo veliko črko, recimo ji *c2*, si zapomnimo
    naslednik[*c1*] = *c2*;
  - Na koncu le sledimo povezavam v tabeli naslednik, da rekonstruiramo celo pot
    kraj = *zacetniKraj*; **while** (*kraj* != –1) *kraj* = *naslednik*[*kraj*];

1.4 Dva od petih

1	11000
2	10100
3	10010
4	01010
5	00110
6	10001
7	01001
8	00101
9	00011
0	01100

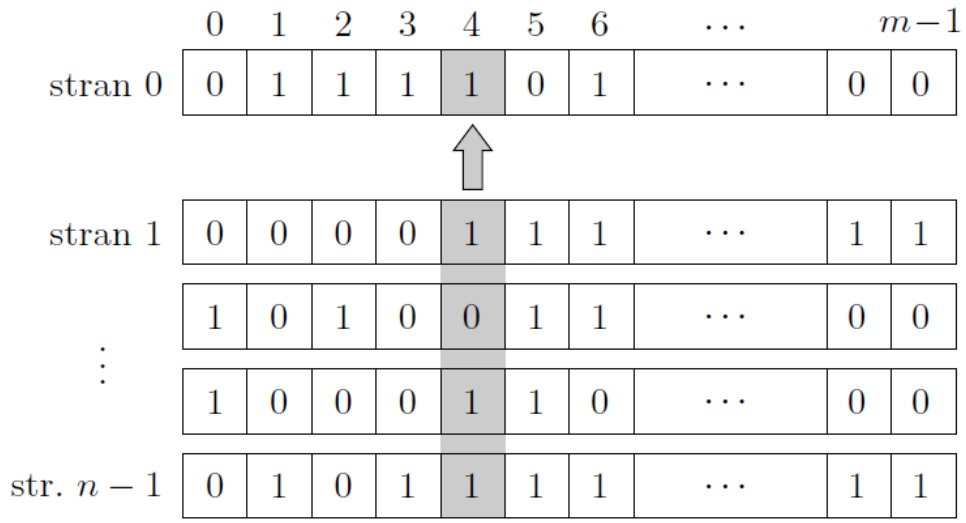
- Števke 0..9 so zakodirane s petericami ničel in enic
- Naloga: preberi vrstico z 10 takimi petericami in jih dekodiraj
  - Primer: 11000 10001 11101 01a1001001 → 16*07
- Rešitev:
  - Berimo podatke znak po znak
  - Vse znake, ki niso 0 ali 1, ignoriramo
  - Ničle in enice si zapomnimo, zraven še štejemo, koliko jih že imamo
    $x = (x << 1) \mid naslednjiBit$; $stBitov = stBitov + 1$
  - Ko stBitov doseže 5, primerjamo x z vsemi desetimi kodami in pogledamo, s katero se ujema (če z nobeno, izpišemo *)
    - Nato postavimo *x* in *stBitov* spet na 0

1.5 Kontrolne vsote



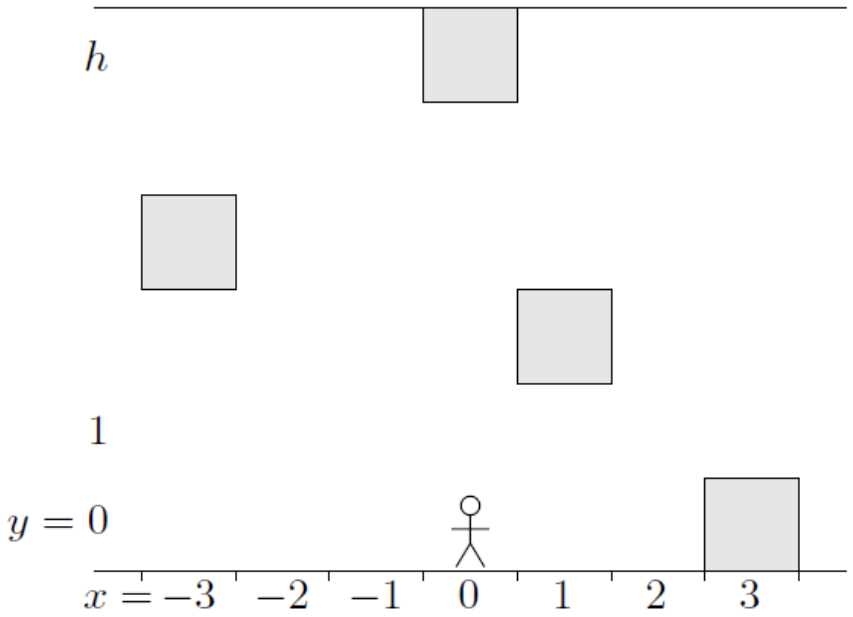
- Imamo pomnilnik, *n* strani × *m* bitov, na strani 0 so kontrolne vsote
  - Kontrolna vsota je 1, če je v stolpcu pod njo liho mnogo enic, sicer pa 0
  - Dani sta funkciji *Beri*(*stran*, naslov) in *Pisi*(*stran*, *naslov*, *novaVrednost*) za dostop do posameznih celic
- Napisati moramo funkciji *Beri2* in *Pisi2*, ki naj skrbita tudi za kontrolne vsote
  - *Pisi2* mora zapisati novo vrednost in popraviti kontrolno vsoto
  - *Beri2* mora prebrati vrednost, če pa branje spodleti zaradi okvare, jo mora rekonstruirati iz kontrolne vsote in ostalih celic v istem stolpcu

1.5 Kontrolne vsote



- Rešitev: *Beri2*
  - Najprej kličemo *Beri* – če ta uspe prebrati celico, vrnemo njeno vrednost in smo končali
  - Sicer preberemo še vse ostale celice v istem stolpcu (vključno s kontrolno vsoto)
  - Če kakšno od teh branj spodleti, obupamo
  - Sicer je xor prebranih bitov ravno enak tistemu v naši celici
- Rešitev: *Pisi2*
  - Preberemo staro vrednost celice, zapišemo novo
    - To je koristno brati z *Beri2*, ne *Beri*
  - Če sta različni, preberemo staro kontrolno vsoto, jo zatogglamo in zapišemo nazaj

2.1 It’s raining cubes

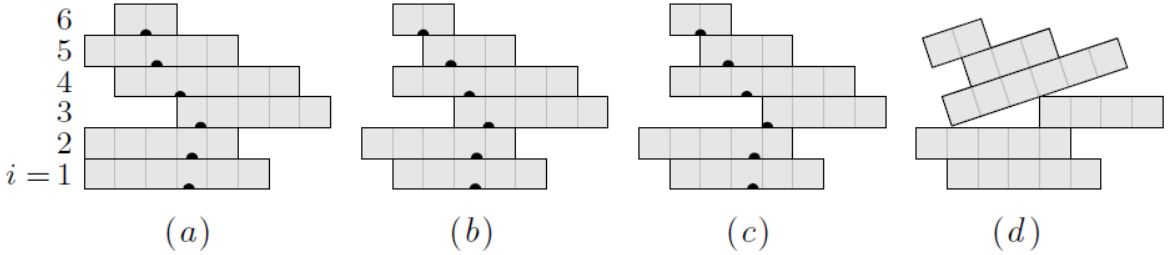


- Stojimo na premici in se premikamo v diskretnih korakih
- S stropa padajo kocke, ob času *i* se pojavi kocka na (x_i, h) in h korakov kasneje prileti na tla
  - Če pade na nas, nas ubije
  - Sicer pa nam le blokira pot, mimo nje (ali čeznjo) ne moremo
- Ali lahko preživimo (in kako)?

2.1 It’s raining cubes

- Lahko se omejimo na poti, ki gredo najprej nekaj časa v eno smer, nato se ustavijo in odtlej stojijo pri miru
  - Nima se smisla ustavljati in nato nadaljevati poti
  - Nima smisla spreminjati smeri
  - Vprašanje je le še, na kateri x-koordinati naj se ustavimo
- Kocka, ki začne padati na x_i ob času i, bo pristala ob času $h + i$
  - Mi pa lahko pridemo na x_i najprej ob času $|x_i|$
  - Torej, če je $|x_i| \geq h + i$, nam bo ta kocka zaprla pot
    - Do polja x_i in vseh nadaljnjih v tisti smeri
- Iz teh omejitev dobimo nek interval dosegljivih polj
  - Nato le še preverimo, če je med njimi kakšno tako, na katero nikoli ne pade nobena kocka

2.2 Strahopetni Hektor

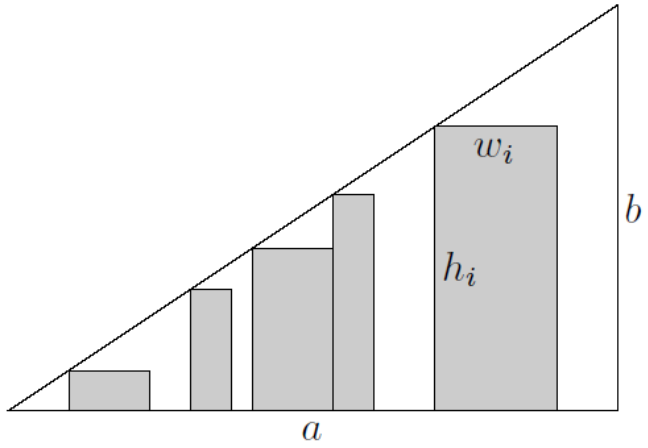


- Imamo zid, sestavljen iz vodoravnih blokov, zloženih eden na drugega
- Podana je začetna oblika zidu, nato pa ga obstreljujemo
  - Vsak strel pride z leve na neki dani višini in uniči skrajno levi kvadratek v tisti vrstici
  - Če se zgodi, da blok v neki vrstici *y* ne podpira več težišča blokov (*y*+1)…*n*, se zgornji del zidu nagne in podre
  - Po vsakem strelu moramo preveriti, ali se je zid že podrl ali ne

2.2 Strahopetni Hektor

- Težišče neke skupine blokov je povprečje *x*-koordinat vseh kvadratkov v teh blokih
  - Naj bo b_y skupno število kvadratkov v blokih $y..n$, s_y pa vsota njihovih koordinat
  - Potem je $t_y = s_y / b_y$
  - Vse to lahko računamo v zanki:
    - Na vrhu začnemo z $b_{n+1} = 0$, $s_{n+1} = 0$
    - Če blok y pokriva kvadratke z_y, $z_y + 1$, …, $z_y + d_y - 1$, je $b_y = b_y + 1 + d_y$, $s_y = s_y + 1 + z_y + (z_y + 1) + \ldots + (z_y + d_y - 1)$
  - Ko pride nov strel v blok *y*:
    - Za vsak $i \leq y$ se b_i zmanjša za 1, s_i pa za z_y
    - Nato povečamo z_y za 1 in zmanjšamo d_y za 1
  - Blok *y* podpira težišče tistih nad sabo, če je $z_y - ½ \leq t_{y+1} \leq z_y + d_y - ½$

2.3 Polaganje plošč



- V pravokotni trikotnik $a \times b$ polagamo pravokotne plošče
  - i-ta plošča je široka w_i in visoka h_i
  - Položimo jo lahko le tako, da se dotika hipotenuze
  - Plošče se ne smejo prekrivati, radi bi jih položili čim več
- Rešitev: požrešni algoritem
  - Levi rob plošče i mora biti na $a \cdot h_i / b$, desni pa zato na $d_i := a \cdot h_i / b + w_i$
  - Pregledujmo plošče po naraščajočem d_i
  - Če se plošča prekriva z dosedanjimi, jo zavrzimo, sicer jo položimo

00011, 00101, 00110, 01001, 01010, 01100, 10001, 10010, 10100, 11000
00000, 00011, 00110, 01001, 01100, 01111, 10010, 10101, 11000, 11011

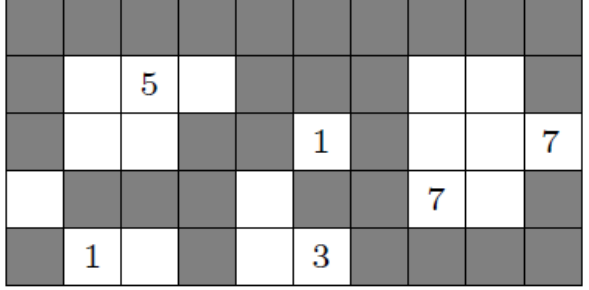
2.4 Kodiranje

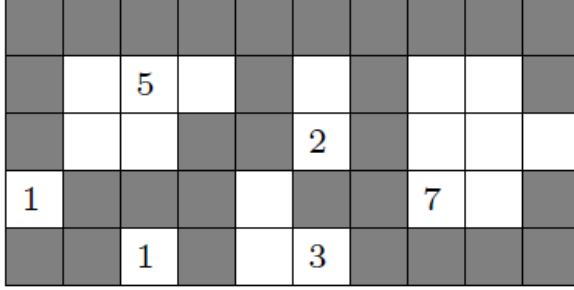
- Danih je 10 peteric bitov
- Ali se lahko zgodi, da če v eni peterici zamenjamo dva sosednja bita, dobimo eno od ostalih 9 peteric?
- Rešitev:
  - Z dvema gnezdenima zankama gremo po vseh parih peteric
  - Za vsak par moramo preveriti, če se peterici dovolj razlikujeta:
    - Primerjamo istoležne bite
    - Če opazimo neujemanje:
      - Če se tega neujemanja ne da odpraviti z zamenjavo dveh sosednjih bitov, se peterici dovolj razlikujeta
      - Če smo že prej izvedli zamenjavo, se peterici dovolj razlikujeta

2.5 Golovec

- Sistem nam sporoči, ko kakšno vozilo pripelje v predor ali zapelje iz njega
  - Dobimo čas in registrsko številko
  - Ob izstopu moramo preveriti, če je bila povprečna hitrost previsoka
- Rešitev:
  - Naloga pravi, da bo v predoru največ 300 vozil hkrati – hranimo jih v tabeli parov (registrska številka, čas vstopa)
  - Ob vsakem klicu pogledamo, če je trenutna registrska številka v tabeli
    - Če ne, gre za vstop: vozilo le dodamo v tabelo
    - Sicer gre zdaj za izstop: izračunamo povprečno hitrost in če je previsoka, opozorimo; vozilo tudi pobrišemo iz tabele

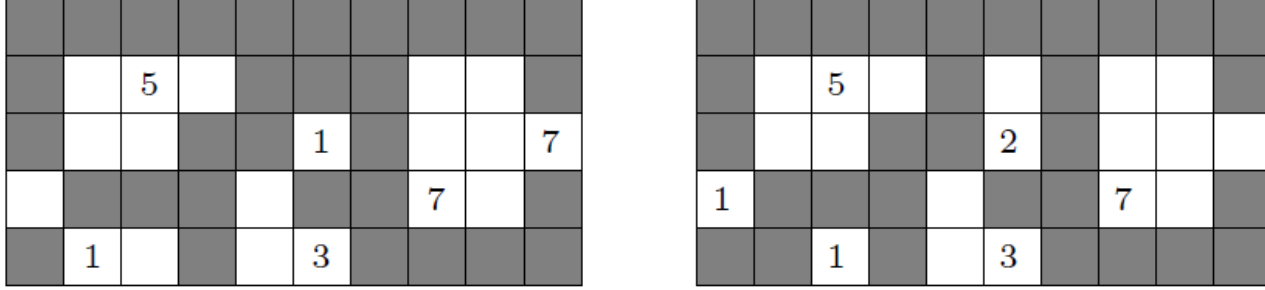
3.1 Nurikabe





- Pravokotna karirasta mreža, vsako polje je črno ali belo
  - Črna polja tvorijo morja, bela tvorijo otoke
  - Če imata polji skupno stranico in sta enake barve, pripadata istemu morju/otoku
  - Na belih poljih so lahko tudi števila
  - Otok je pravilno označen, če je na njem eno samo število in je to število enako površini otoka
  - Naloga: preštej morja, otoke, pravilno označene otoke, črne kvadrate 2 × 2

3.1 Nurikabe



- Rešitev:
  - Črni kvadrati 2 × 2: gremo v dveh zankah po vseh poljih in pogledamo, če je polje + 3 sosedi črni
  - Morja/otoki: začnemo v poljubnem polju in preiskujemo npr. v širino (*breadth-first search*)
    - Dodamo polje v vrsto
    - Dokler vrsta ni prazna:
      - Vzamemo neko polje iz vrste
      - Pregledamo njegove sosede
      - Če pripadajo istemu morju/otoku, jih dodamo v vrsto
    - Ko neko polje dodamo v vrsto, ga označimo, da ga kasneje ne bomo dodali še enkrat
    - Na koncu poznamo površino morja/otoka, spotoma gledamo tudi, če so na njem kakšne številke (in na koncu preverimo, če je pravilno označen)
  - To ponavljamo, dokler ni pregledana cela mreža

3.2 Analiza signala

- Imamo več oddajnikov, ki oddajajo ničle in enice
  - Vsak oddajnik ima konstantno periodo – na vsakih toliko korakov odda enico, vmes pa ničle
  - Npr. en oddajnik ima periodo 2: 1 0 1 0 1 0 1 0 1 0 …
  - Nek drug pa periodo 3: 1 0 0 1 0 0 1 0 0 1 …
  - So sinhronizirani – na začetku vsi oddajo enico
  - Mi ne vemo, koliko je oddajnikov in kakšne periode imajo
  - Dobimo le vsoto signalov: npr. 2 0 1 1 1 0 2 0 1 1 …
  - Naloga: povej, koliko je oddajnikov s kakšno periodo

3.2 Analiza signala

- Označimo našo vsoto signalov z $a_0\ a_1\ a_2 \ldots a_{n-1}$
  - Ob času 0 vsak oddajnik odda enico → a_0 je kar število vseh oddajnikov
  - Poiščimo prvi naslednji neničelni element,
    torej najmanjši tak $t > 0$, pri katerem je $a_t > 0$
    - Ob času t oddajo enico le oddajniki, katerih perioda je delitelj t-ja
    - Toda noben oddajnik nima periode, manjše od t (ker so $a_1 = \ldots = a_{t-1} = 0$)
    - Torej imamo a_t oddajnikov s periodo t; to si zapomnimo v neki tabeli
    - To število oddajnikov odštejmo od a_0, a_t, a_{2t}, a_{3t}, …
  - Gornji postopek ponavljajmo, dokler niso vsi $a_1 = \ldots = a_{n-1} = 0$
  - Če je zdaj a_0 še vedno > 0, vemo, da imamo še toliko oddajnikov
    s periodami ≥ n (pri katerih periode ne moremo točno določiti)

3.3 Mafijski semenj

1	2	5
1	4	6
2	4	4
2	3	6
1	2	1
2	5	2
1	5	6

- Imamo n mafijcev, i-ti od njih pripada združbi z_i in uperi pištoli v mafijca l_i in d_i
  - Tisti, na katerega ne meri noben mafijec iz kakšne druge združbe, odide
  - To se ponavlja, dokler je mogoče
  - Kdo na koncu ostane?
- Iz tega lahko naredimo graf:
  - Po ena točka za vsakega mafijca
  - Usmerjena povezava $u \rightarrow v$, če u meri s pištolo na v in pripadata različnima združbama
  - Rešitev je enak postopek kot pri topološkem urejanju grafa

3.3 Mafijski semenj

- Določimo vhodno stopnjo vsake točke
  (s koliko pištolami kažejo nanj mafijci iz drugih združb?)
  - Na začetku postavimo vse vhodne stopnje na 0
  - Za vsako povezavo $u \to v$: povečaj vhodno stopnjo v-ja za 1
- Tisti, ki imajo zdaj vhodno stopnjo 0, odidejo – dodajmo jih v vrsto
- Dokler vrsta ni prazna:
  - Vzemi naslednjega mafijca (recimo mu u) iz vrste
  - Za vsako povezavo $u \to v$:
    - Zmanjšaj vhodno stopnjo v-ja za 1
    - Če je padla na 0, dodaj v-ja v vrsto
- Na koncu izpiši tiste, ki imajo še vedno stopnjo > 0

3.4 Trgovanje z zrni

- Dana so števila $a_1, a_2, \ldots, a_n$ ($n \leq 40$)
  - Radi bi jih nekaj izbrali tako, da bo njihova vsota čim bližje q (ne pa večja!)
  - Možnih vsot je 2^n, kar je preveč (2^{40} = več kot 1 bilijon)
- Rešitev: razdelimo jih v dve skupini, $\{a_1, \ldots, a_k\}$ in $\{a_{k+1}, \ldots, a_n\}$
  - Za vsako skupino pripravimo urejen seznam vseh možnih vsot (do 2^k vsot pri prvi skupini in do 2^{n-k} v drugi)
  - Naj bo prvi seznam $v_1, \ldots, v_s$, drugi pa $u_1, \ldots, u_t$
  - Za vsako vsoto v_i iz prvega seznama moramo zdaj v drugem seznamu poiskati največji tak u_j, za katerega je $v_i + u_j$ še $\leq t$
  - To lahko počnemo npr. z bisekcijo ali pa z zlivanjem

3.4 Trgovanje z zrni

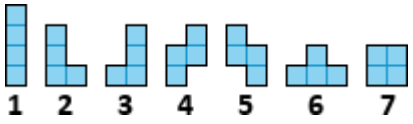
- Zlivanje:
  - Postavimo se na začetek prvega seznama (i = 1) in na konec drugega ($j = t$)
  - Dokler je $v_i + u_j > t$, zmanjšujemo j za 1 (*)
  - Zdaj imamo vsoto $v_i + u_j \leq t$; če je večja od največje vsote doslej, si jo zapomnimo
  - Premaknemo se naprej po prvem seznamu ($i = i + 1$) in gremo na (*)
- Zakaj to deluje?
  - Če so pri nekem i prevelike vse vsote $v_i + u_{j'}$ za $j' > j$, so pri $i + 1$ vsote $v_{i+1} + u_{j'}$ za $j' > j$ še toliko bolj prevelike (ker je $v_{i+1} > v_i$)
  - Zato, ko se i poveča, se lahko j le zmanjša ali ostane enak, ne more pa se povečati
- Časovna zahtevnost takega zlivanja je $O(2^k + 2^{n-k})$
  - Vzemimo $k = n/2$; pri $n = 40$ imamo $k = 20$, zlivamo dva seznama dolžine ≈1 milijon

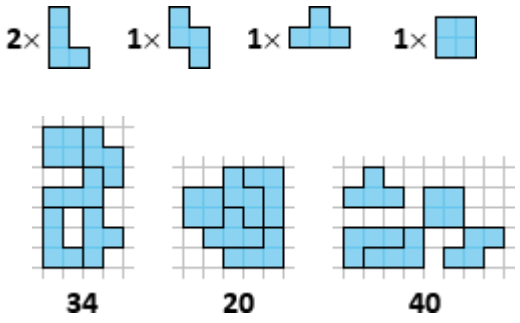
3.5 Razcep niza

- Imamo nize ničel in enic
  - Definirajmo oceno niza s kot $f(s) = \min\{ \textit{štNiče}l(s), \textit{štEnic}(s) \}$
  - Dani niz $a = a_1\, a_2 \ldots a_n$ hočemo razbiti na k nepraznih podnizov tako, da bo vsota njihovih ocen čim manjša
    - Primer: a = 110100100, k = 3 → 1101 + 001 + 00 (ocena: 1 + 1 + 0 = 2)
- Rešitev: dinamično programiranje
  - V najboljšem razbitju a-ja na k podnizov je zadnji podniz oblike $a_{m+1}\, a_{m+2} \ldots a_n$ za nek $m < n$
  - Pred tem imamo torej najboljše razbitje niza $a_1 \ldots a_m$ na k – 1 podnizov
  - To je problem enake oblike kot prvotni, le niz je krajši in zahtevan je 1 podniz manj
  - Naj bo $g(m, t)$ ocena najboljšega razbitja niza $a_1 \ldots a_m$ na t podnizov
  - Zdaj imamo torej $g(m, t) = \min \{ g(r, t-1) + f(a_{r+1} \ldots a_m) : t - 1 \le r < m \}$
  - To lahko računamo sistematično po naraščajočih t, na koncu vrnemo $g(n, k)$

Off-line naloga – Zlaganje likov

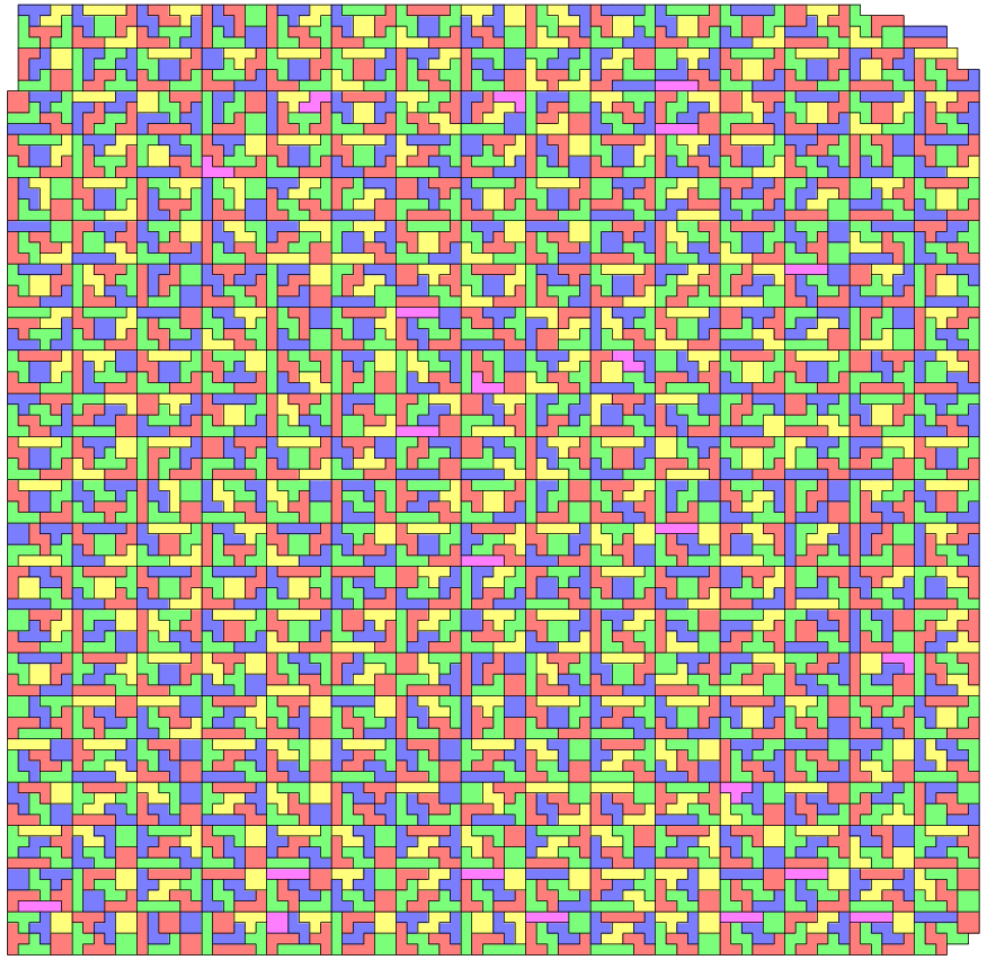
- Danih je več likov iz igre Tetris
  - Možnih je 7 vrst likov
  - Vsake vrste dobimo po največ 300 likov
- Radi bi jih zložili v večji lik s čim manjšim obsegom
  - Seveda se ne smejo prekrivati
- 300 testnih primerov
- Čas za reševanje: od septembra 2014 do vključno 20. marca 2015
  - Zadnja oddaja je prišla 20. marca ob 23:59:33 :)

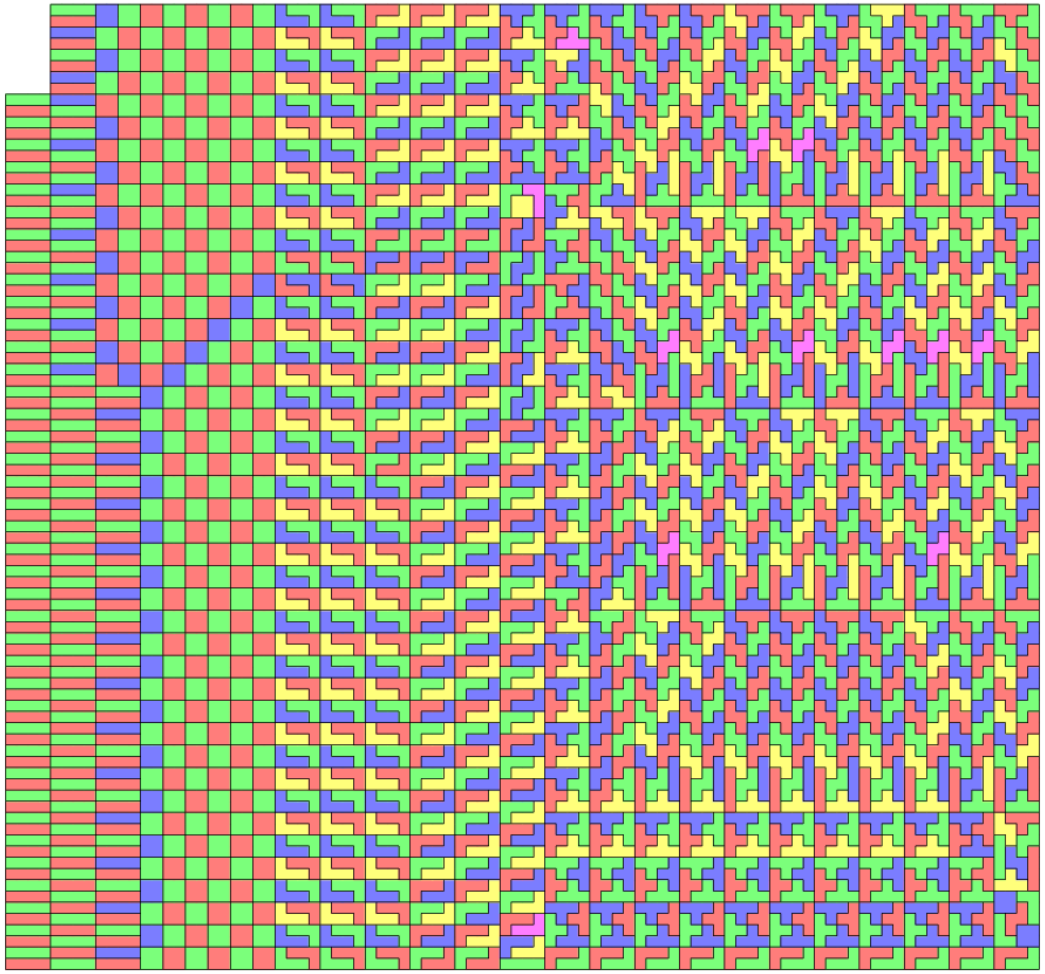


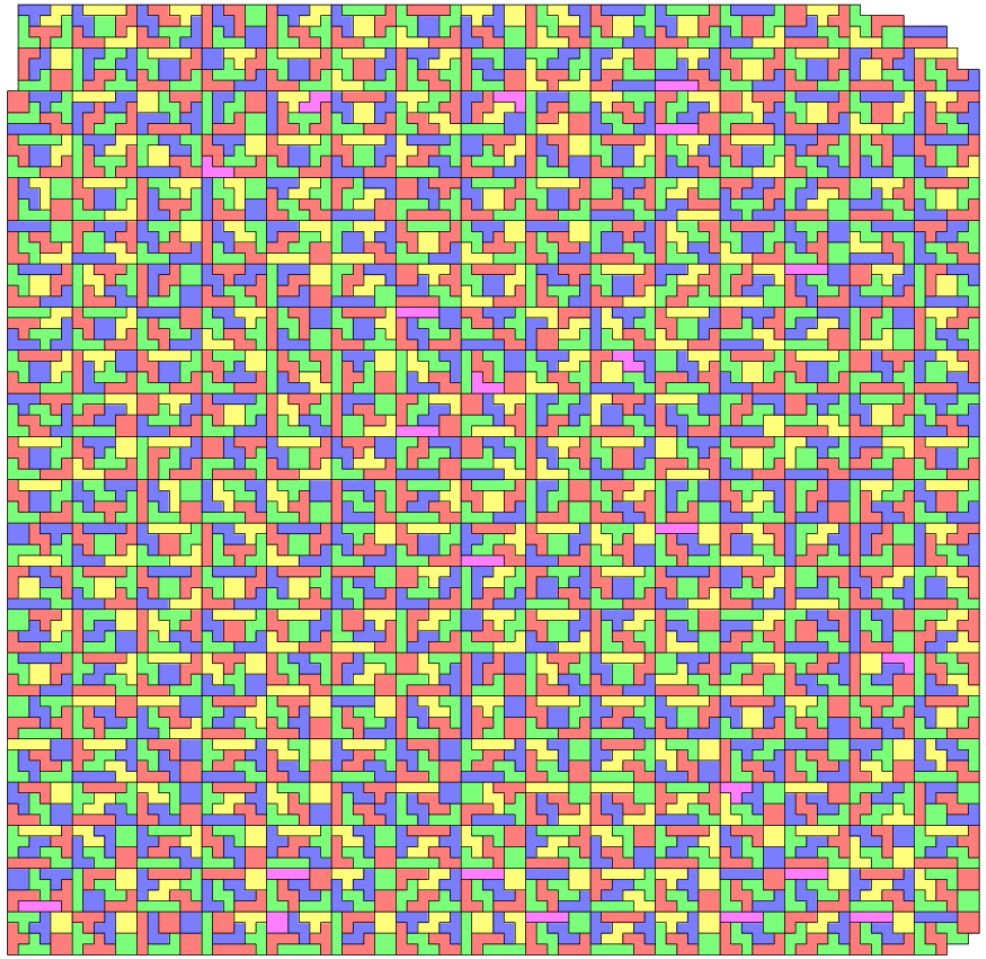
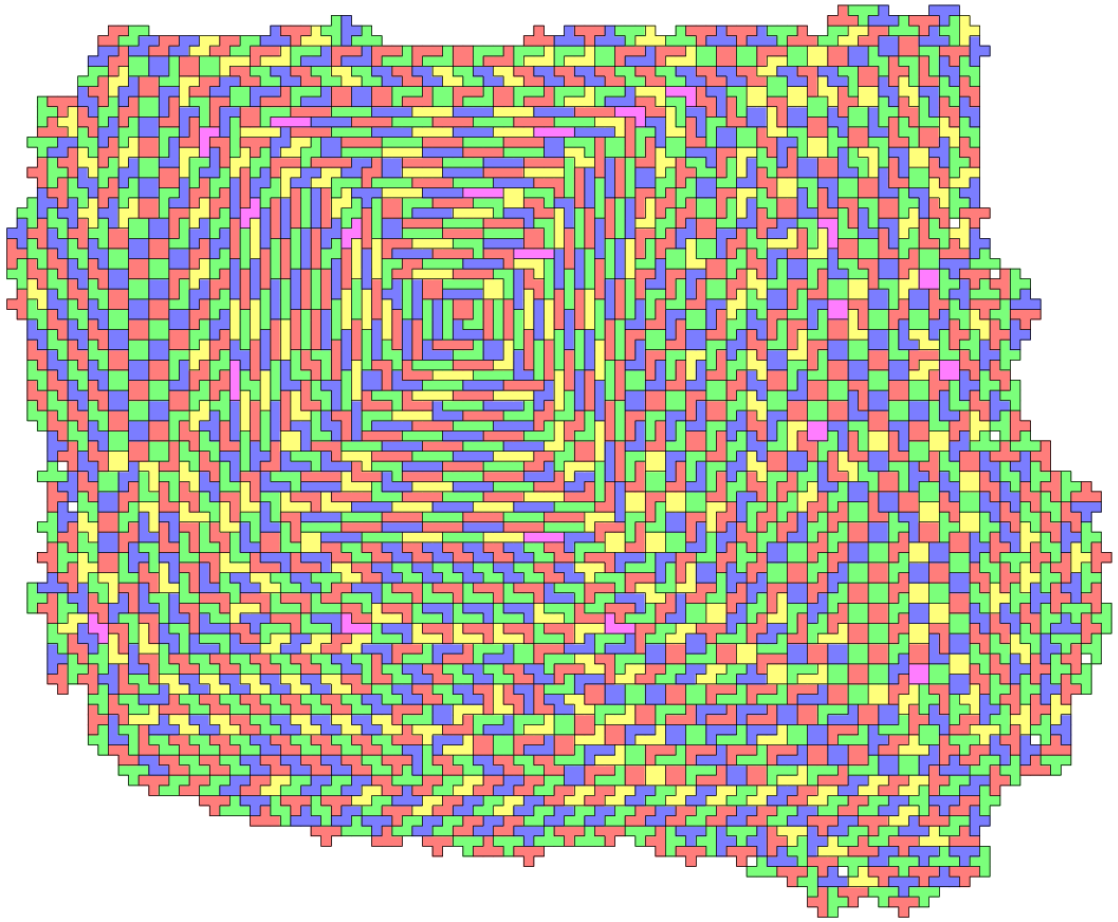


Sistematično zlaganje

- Dve različni, a enako dobri rešitvi istega testnega primera (#233)
  - Obe imata obseg 356





Požrešni algoritem

- Še vedno isti testni primer, na desni je požrešna rešitev
  - Njen obseg je kar 502