Varianta 3

1.

a) Corectitudine:

Se calculează sumele de pe poziţiile pare, respectiv impare, ale şirului iniţial:

$$S_p = \sum_{i=1}^{n/2} a_{2i}$$

$$S_i = \sum_{i=1}^{n/2-1} a_{2i+1}$$

Cum dimensiunea ~~vechi~~ şirului este întotdeauna pară, la pasul întâi (prima mutare a jucătorului 1) şirul va avea forma:

$a_1 \; a_2 \; \ldots \; a_n$

Cazul I: jucătorul 1 alege stânga. Şirul devine:

$a_2 \; \ldots \; a_n$

Cazul II: jucătorul 1 alege dreapta. Şirul devine:

$a_1 \; \ldots \; a_{n-1}$

Prin urmare jucătorul 1 compară S_p şi S_i

1° Dacă $S_p \geq S_i$, jucătorul 1 va alege de fiecare dată elementul de index par, astfel forţându-l pe jucătorul 2 să aleagă dintre 2 indecşi impari.

2° Dacă ~~$S_i < S_p$~~ $S_p < S_i$, jucătorul 1 forţează jucătorul 2 să aleagă între indecşi pari, alegând de fiecare dată indexul impar.

Deoarece la fiecare mișcare a jucătorului 1 vrul e de forma:

$a_{2i} \dots a_{2j+1}$ sau $a_{2i+1} \dots a_{2j}$

acesta poate forța jucătorul doi să aleagă doar între indici de aceeași paritate.

Prin urmare, la final:

$S_{j1} = \sum_{i=1}^{n/2} a_{2i}$, $S_{j2} = \sum_{i=1}^{n/2-1} a_{2i+1}$, dacă $S_p \geq S_i \Rightarrow$

$\Rightarrow J_1$ câștigă

$S_{j1} = \sum_{i=1}^{n/2-1} a_{2i+1}$, $S_{j2} = \sum_{i=1}^{n/2} a_{2i}$, dacă $S_p < S_i \Rightarrow$

$\Rightarrow J_1$ câștigă

b) Deoarece strategia implementată la punctul a) este optimă, egalitatea având loc doar atunci când $S_p = S_i$, nu există altă strategie care să ofere câștigul garantat cu $S_{j1} > S_{j2}$.

Deși pot exista alte strategii cu care J_1 ar putea câștiga cu o mică doză de noroc, acestea ar presupune ca J_2 să nu joace "optim", ceea ce nu este garantat.

(2.) Corectitudine:

Algoritmul realizează o parcurgere în adâncime și colorează nodurile cu negru sau alb.

Corectitudinea este dată de proprietatea arborilor de a putea fi colorați în două culori a.î. nici o pereche de noduri adiacente ~~se~~ nu va avea aceeași culoare.

Prin urmare, trebuie doar să numărăm nodurile colorate cu alb și ele cu negru și să alegem numărul mai mare.

Nodurile colorate cu culoarea majoritară vor fi elementele mulțimii independente maximale.

O metodă similară nu va funcționa pentru un graf neorientat oarecare deoarece nu ~~se mai~~ garantează o 2-colorare corectă a respectivului graf.

(3)

a) Soluția generată de algoritm e corectă deoarece toate intervalele sunt adăugate într-o submulțime și prin modul de alegere a submulțimilor, ~~intervalele~~ un interval este adăugat la o submulțime doar dacă nu se intersectează cu un alt interval din submulțime.

Numărul minim de partiții necesare este egal cu adâncimea lui S, unde adâncimea reprezintă numărul maxim de intervale care se suprapun peste un același punct.

Dem

Notăm cu d adâncimea lui S ⇒
⇒ ∃ o mulțime de intervale {I1, ..., Id}
ce se suprapun peste un același punct.
Deoarece nu putem pune în aceeași submulțime intervale care se suprapun ⇒
⇒ I1, ..., Id se pun în submulțimi (partiții) diferite ⇒ avem nevoie de cel puțin d partiții.

~~Fie k nou~~

Fie k_n nr. de partiții folosite la pasul curent. Inițial $k_n = 0$

Fie I_j intervalul care trebuie adăugat unei partiții la pasul curent

Dacă I_j se poate adăuga unei partiții existente, atunci k_n nu se modifică.

Dacă I_j nu se poate adăuga niciunei partiții existente, atunci $\exists$ k_n intervale din partiții diferite 2 câte 2, cu extremitatea inițială mai mică sau egală decât a lui I_j ce se intersectează cu acesta.

Deoarece toate cele k_n intervale au extremitatea inițială $\leq$ cea a lui I_j $\Rightarrow$ $\exists$ cel puțin un punct în care toate cele k_n+1 intervale se suprapun, anume extremitatea inițială a lui I_j. $\Rightarrow$

$\Rightarrow k_n + 1 \leq d \Rightarrow$ algoritmul nu va folosi mai mult de k_n partiții.

Prin urmare algoritmul găsește o partiționare optimă.

b) Contraexemplu:

Pentru intervalele: (4,6); (4,7); (7,10); (6,10)

Soluția găsită de algoritm:

1: (4,6) (7,10)

2: (4,7)

3: (6,10)

Soluția optimă:

1: (4,6) (6,10)

2: (4,7) (7,10)

c) Corectitudine:

Considerăm k numărul de partiții folosite la pasul curent.

Fie I_j intervalul care trebuie adăugat unei partiții la pasul curent.

Dacă I_j nu se poate ~~aduga~~ adăuga unei partiții existente $\Rightarrow$ $\exists$ k intervale distincte 2 câte 2 ce se intersectează cu I_j și au extremitatea finală $\leq$ cea a lui I_j. Considerăm pe cea mai mică extremitate finală dintre cele ale ultimelor intervale adăugate în fiecare partiție. $\Rightarrow$

$\Rightarrow$ $\exists$ un interval ce conține pe în fiecare

partiție, $u \in I_j \Rightarrow \exists h+1$ intervale care se suprapun în $u \Rightarrow h+1 \leq d$

Analog punctului a) $\Rightarrow$ algoritmul găsește soluția optimă.

d) Analog b)

e) Analog c)