Vars Pb1

b) Da este posibil. Putem încerca să alegem elementul cel mai mare la mom. respectiv, dacă suma termenilor de pe poz. impare este egală cu suma termenilor de pe poz. pară. Ex.:

1 3 4 2 -> alegem 2

↓

• 1 3 4 -> calc. alege 1

↳ 3 4 -> alegem 4 (4>3)

3 - alege calc.

-> Suma_player1: 2+4 = 6
Suma_player2: 3+1 = 4 | => câștigăm

↳ dacă calc. alege 4 atunci o să iasă egal mereu.

Var 3 Pb 3

a) Teorema: În orice instanță a pb. nr. de ~~săli~~ partiții va fi ~~cel puțin~~ măcar ~~m.~~ nr. de intervale.

Dem: Avem d intervale, $I_1, I_2, \dots, I_d$ cu proprietatea că toate au un punct comun ~~Pt~~ => vom crea o partiție pt fiecare interval în parte deci vom avea ~~măcar~~ cel puțin d ~~săli~~ partiții.

- Varianta este optimă ($O(n \log n)$)

· Partiția d este creată pt. că intervalul j se suprapune cu celelalte d-1.

· Din mom. ce intervalele sunt sortate după timpul de start, orice suprapunere apărută este cauzată de intervale care încep mai devreme. (start pt. intervalul j)

· Prin urmare, avem d-1 de intervale care se suprapun la $s_j + \varepsilon$ (unde $\varepsilon$ este f. mic). Acest lucru este valabil ori de câte ori deschidem o ~~sală~~ partiție.

În concluzie, greedy construiește d partiții și d este minimul de care avem nevoie.

B) Exemplu:

(1,2)(1,3)(1,4)(2,4),(3,5)(4,6)(4,6)(6,8)
(6,7)(6,8)

1: (1,2)(2,4)(4,6)(6,8)
2: (1,3)(3,5)(6,7)
3: (1,4)(4,6)(6,8)

b) Această variantă nu este bună deoarece
în anumite cazuri eșuează.

~~(1,3)~~, (1,2)(1,3)(3,7)(2,7)

1: (1,2)(3,7)   Fals   1: (1,2)(2,7)
2: (1,3)   $\Rightarrow$ 2: (1,3)(3,7)
3: (2,7)

~~c) Este corect dar va crește timpul de execuție~~
~~pt. că, în plus, mai face~~

d) Nu va funcționa deoarece în pb. spec.
tacalilor intervalele sunt sortate cresc. după
timpul de sf $\Rightarrow$ o să ajungem în cazul
b)