Piraţii din Caraibe – Mihai Pătraşcu

Ideea este să raţionăm pornind de la ultimul pirat spre primul. Daca **N-2** piraţi sunt aruncaţi peste bord, piratul **N-1** îşi va acorda întreaga comoară lui. Piratul **N** nu poate să îl împiedice, pentru că **A[N-1] = 0**. Deci piratul **N-3** ştie ce se va întâmpla dacă el va fi aruncat peste bord (**N-1** primeşte tot, şi **N** primeşte zero). Bazat pe asta, el poate propune o schemă care îi maximizează profitul şi este acceptată. Deci acum piratul **N-4** ştie precis ce se va întâmpla dacă el este aruncat peste bord. Similar, prin inducţie, piratul **N-i-1** ştie ce se va întâmpla dacă el este aruncat peste bord (va fi adoptată schema piratului **N-i**), şi poate lua o decizie bazată pe această certitudine.

Când propune o schemă, un pirat **J** vrea să îşi maximizeze profitul, dar este necesar ca schema lui să fie acceptată (altfel va fi aruncat peste bord). Pentru asta, el trebuie la **A[J]** piraţi să dea mai mulţi bănuţi decât aceştia ar primii în schema piratului **J+1**. Pentru a-şi maximiza profitul, el alege să convingă în acest mod pe piraţii care primesc cele mai mici sume în schema lui **J+1**. La aceste sume, el adaugă câte un bănuţ. La ceilalţi piraţi, el le dă zero pentru că are deja suficienţi susţinători. Algoritmul va fi în esenţă acesta: considerăm toţi **J** de la **N-2** până la **1**; la pasul **J** ştim deja schema pe care ar propune-o **J+1**; pentru a afla schema piratului **J**, alegem **A[J]** piraţi cu cele mai mici sume în schema precedentă, incrementăm câştigul acestor piraţi, şi facem câştigul celorlalţi egal cu zero.

Se observă că măcar suma unui pirat va fi facută zero (pentru că **A[J] < N-J**). Deci suma pe care o câştiga **J** este cel puţin suma maximă din suma lui **J+1** minus **A[J]** (fiindcă trebuie incrementate **A[J]** sume). Pentru că numărul de bănuţi este foarte mare, reiese uşor că suma maximă din schema lui **J** este ce îşi acordă **J** lui însuşi. Mai precis, în schema proprie **J** câştigă cel puţin 10 miliarde **–A[N-2] –A[N-1] – ... –A[J+1]**. Deci scădem maxim **N(N+1)/2 <** 5 miliarde şi suma rămasă este mai mult de jumătate din bănuţi. Asta înseamnă că orice **J** poate propune o schemă în care să nu fie aruncat peste bord, şi mai mult poate câştiga mai mult decât orice pirat rămas.

Implementarea algoritmul este destul de uşoară şi are un timp de rulare **O(N)**. Ţinem o coadă cu piraţii în ordinea crescătoare a sumelor câştigate. Dacă mai mulţi piraţi câştigă aceaşi sumă, ţinem în coadă un singur element (numărul de piraţi care câştigă acea sumă). La pasul **J**, extragem din vârful cozii toţi piraţii care vor primii zero, până rămân doar **A[J]** piraţi. Toţi aceşti piraţi vor fi introduşi la bază cozii ca un singur element (toţi primesc zero). Nu e nevoie să incrementăm explicit sumele pentru ceilalţi, ci putem să observăm că toţi din grupul **I** (unde primul grup din coadă are **I=0**) primesc **I** bănuţi, deci suma e dată de poziţia în coadă (care se schimbă în timp). Este uşor de văzut ca algoritmul rulează în timp **O(N)**: la fiecare pas, în coadă se bagă doar un element (un grup de piraţi cu suma zero), deci numărul total de elemente scoase din coadă pe parcursul rulării este cel mult **N**.