Prima observație este că nu ne interesează ordinea în care au fost date primele N voturi, ci doar câte voturi a primit fiecare candidat. Vom ține această informație într-un vector de frecvențe F. Următoarea observație este că, deoarece N M, va trebui să știm toate elementele acetui vector F pentru a răspunde corect celor M întrebări. Aceste răspunsuri se pot determina în O(1) modificând vectorul de frecvențe cu cele M noi voturi și ținând un maxim.

10 puncte

Putem transmite direct cele K elemente ale acestui vector folosing `K log N` biți.

30 de puncte

Observăm că suma elementelor vectorului F este N, iar vectorul are lungime K. Putem reprezenta acest vector ca un șir binar de N+K-1 elemente, construindu-l astfel:

Pentru fiecare candidat i, punem la finalul șirului un număr de biți de 1 egal cu numărul de voturi primite de candidatul i, apoi un 0 (exceptând situația ultimului candidat, când nu adăugăm 0). De exemplu, pentru vectorul de frecvențe (4, 0, 3, 2) reprezentarea este 111100111011.

Putem transmite acest șir de biți de lungime N+K-1 pentru 30 de puncte.

75 de puncte

Observăm că numărul total de vectori de frecvențe în care suma elementelor este N și numărul de elemente este K este R(N, K) = Combinari(N+K-1, K-1). Această formulă se deduce din modalitatea de construcție a reprezentării pe N+K-1 biți: trebuie să alegem, din cei N+K-1 biți, K-1 pe care să îi facem 0, restul fiind 1.

Vom încerca acum să vedem, dacă am sorta toate cele R(N, K) reprezentări posibile în ordine lexicografică, pe ce poziție P s-ar afla reprezentarea șirului F. Vom face acest lucru în felul următor:

Fie RN și RK numărul de voturi, respectiv numărul de candidați rămași de procesat. Inițializăm RN cu N și RK cu K. Iterăm prin toate elementele șirului nostru. La pasul i, dacă bitul curent este 1, adăugăm la P valoarea R(RN, RK-1), reprezentând numărul de șiruri care au primele (i-1) elemente egale cu cele ale șirului nostru și un sufix care ar începe cu 0, deci ar fi mai mic lexicografic decât sufixul al șirului nostru. Apoi decrementăm RN dacă bitul din reprezentare este 1 (am procesat încă un vot), sau RK dacă bitul din reprezentare este 0 (am trecut la următorul candidat).

Transmitem reprezentarea binară a lui `P`, care are `log R(N, K)` biți.

Pentru a reconstitui șirul, procedăm în mod similar. Inițializăm RN cu N și RK cu K. Iterăm prin biții șirului de reconstruit. La pasul i, dacă R(RN, RK-1) <= P, înseamnă că șirul nostru conține 0 pe poziția i (sunt suficiente prefixe care încep cu 0 care să cuprindă și poziția șirului nostru). Altfel, plasăm 1 pe

poziția i și scădem R(RN, RK-1) din P. Apoi decrementăm RN dacă bitul din reprezentare este 1 (am procesat încă un vot), sau RK dacă bitul din reprezentare este 0 (am trecut la următorul candidat).

Operațiile asupra lui P și calculul lui R(N, K) trebuie făcute pe numere mari. Complexitatea calculării lui R(RN, RK-1) este O(K*NrMari) dacă o executăm cu înmulțiri și împărțiri repetate, astfel încât complexitatea totală este O((N+K)*K*NrMari).

100 de puncte

Observăm că de fiecare dată când calculăm R(RN, RK-1), la pasul următor trebuie să calculăm fie R(RN, RK-2), fie R(RN-1, RK-1). R(N, K) fiind o combinare, putem deduce valoarea următoarei pe care o calculăm doar printr-o înmulțire și apoi o împărțire. Acest lucru reduce calculul R(RN, RK-1) la O(NrMari), deci complexitatea totală devine O((N+K)*NrMari).