

## Task A12. Bits and Tree

 15.00 s  2048 MB

Se dă un șir binar  $S$  de lungime  $2N$ . Sarcina ta este să codifici **cel mai lung prefix** posibil al lui  $S$  folosind un arbore neorientat și neetichetat cu  $N$  noduri.

Pentru a face acest lucru, trebuie să scrii două funcții: un codificator și un decodificator. Codificatorul tău va primi șirul binar și va construi un arbore cu noduri etichetate de la 0 la  $N - 1$ . Totuși, mediul de comunicare este nesigur și va **corupe arborele** adăugând un nod suplimentar și conectându-l la unul dintre cele  $N$  noduri originale. Decodificatorul va primi doar această versiune coruptă a arborelui, care are acum  $N + 1$  noduri și  $N$  muchii. În plus, etichetele, muchiile (și nodurile din fiecare muchie) vor fi permutate aleatoriu înainte de a fi transmise decodificatorului. Aceasta simulează transmiterea unui arbore **neetichetat** către decodificator. Decodificatorul trebuie să se bazeze doar pe structura arborelui și nu poate presupune nimic despre etichetele nodurilor.

Decodificatorul tău trebuie să returneze un șir care să aibă un prefix comun cât mai lung posibil cu șirul original  $S$  **indiferent unde apare coruperea**. Scorul tău va depinde de lungimea minimă a prefixului comun în *toate scenariile posibile de corupere* și în toate subtestele tuturor testelor.

### Detalii de implementare

Trebuie să implementezi următoarele funcții:

`std::vector<std::pair<int, int>> encode(int n, std::vector<bool> data)`

Această funcție primește: un întreg  $N$  (numărul de noduri pe care ai voie să le folosești) și un vector boolean `data`, care reprezintă șirul binar  $S$ . Trebuie să returneze o listă de  $N - 1$  muchii ce formează un arbore cu  $N$  noduri (indexate de la 0).

`std::vector<bool> decode(int n, std::vector<std::pair<int, int>> tree)`

Această funcție primește: un întreg  $N$  (care este valoarea originală transmisă către `encode`) și un vector de  $N$  muchii ce formează un arbore cu  $N + 1$  noduri (versiunea coruptă). Trebuie să returneze o secvență de biți care să se potrivească cu `data` pe un prefix cât mai lung posibil.

Pentru fiecare test dat, evaluatorul va rula 1 instanță a codorului și  $N$  instanțe diferite ale decodorului. Fiecare dintre aceste instanțe va avea funcția sa apelată de  $T$  ori — o dată pentru fiecare subtest al testului.

Programul tău nu trebuie să implementeze o funcție `main` și nu trebuie să interacționeze cu intrarea și ieșirea standard. Va fi compilat împreună cu un system grader care se va ocupa de aceste părți.

### Constrângeri

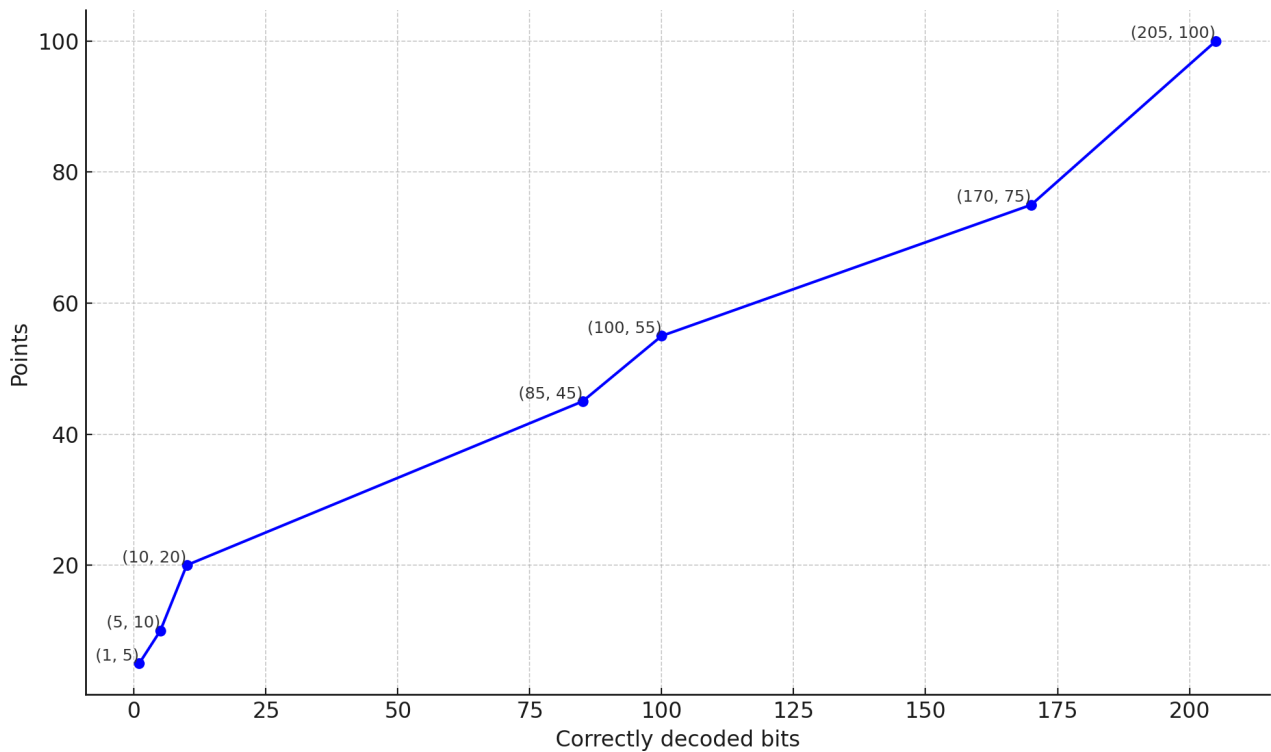
- $T = 2$
- $N = 200$
- $|\text{data}| = 2N$
- Indicii nodurilor variază de la 0 la  $N - 1$  în arborele necorupt.
- Indicii nodurilor variază de la 0 la  $N$  în arborele corupt.

### Punctaj

Punctajul final pentru problemă (ca fracție din punctajul maxim) se bazează pe numărul de biți corect decodificați de soluția ta.

Dacă soluția ta produce un arbore invalid, punctajul va fi 0.

În caz contrar, punctajul tău  $S$  este calculat folosind următoarea funcție liniară pe porțiuni:



Numărul de biți corect decodificați este minimul dintre toate testele, subtestele și scenariile de corupere.

### Testare local 

Este oferit un evaluator local pentru a te ajuta s   ti testezi solu ia. Acesta trebuie compilat  mpreun  cu programul t u. Nu folose te procese separate.

Simuleaz   ntregul flux de codificare/decodificare:

1. Cite te un  ntreg  $T$  — num rul de subteste.
2. Pentru fiecare test:
  - (a) Cite te un  ntreg  $N$  — num rul de noduri disponibile pentru codificare.
  - (b) Cite te un  ntreg  $\ell$  — lungimea  irului de date binare (de obicei  $2N$ ).
  - (c) Cite te  $\ell$  bi i (fiecare 0 sau 1, f r  spa ii) care reprezint   irul binar  $S$ .
  - (d) Apeleaz  func ia ta `encode(n, data)`.
  - (e)  ncearc  s  corup  arborele rezultat prin:
    - Iterarea prin fiecare  $x$  — indicele nodului (de la 0 la  $N-1$ ) la care se ata eaz  un nou nod.
    - Ad ugarea unui nou nod  i ata area la nodul  $x$ .
    - Permutarea aleatoare a etichetelor nodurilor, amestecarea listei muchiilor  i inversarea aleatoare a capetelor fiec rei muchii.
  - (f) Apeleaz  func ia ta `decode(n, corrupted_edges)` cu fiecare arbore corupt.
  - (g) Compar  fiecare rezultat cu datele originale  i num r  c  i bi i se potrivesc  n cel mai scurt prefix comun.

### Exemplu de Interac iune

Input	Interac�iune
1 5 10 1101010110	<code>encode(5, {1,1,0,1,0,1,0,1,1,0})</code> returneaz� <code>{{0,1},{1,2},{0,3},{1,4}}</code> <code>decode(5, {{1,0},{0,2},{2,3},{3,4},{5,3}})</code> returneaz� <code>{1,1,0}</code> <code>decode(5, {{1,0},{0,2},{0,3},{4,0},{5,4}})</code> returneaz� <code>{1,1,1,0,1,0,1,0}</code> <code>decode(5, {{2,0},{2,1},{1,3},{4,1},{5,4}})</code> returneaz� <code>{1,1,1,0,0,0,0,0}</code> <code>decode(5, {{0,1},{0,2},{0,3},{4,3},{3,5}})</code> returneaz� <code>{1,1,1,0,1}</code> <code>decode(5, {{1,0},{1,2},{2,3},{4,2},{5,4}})</code> returneaz� <code>{1,1,1,0}</code>