

Task A12. Bits and Tree

🕒 15.00 s 📁 2048 MB

Даден е двоичен низ S с дължина $2N$. Вашата задача е да кодирате възможно **най-дългия префикс** на S , използвайки неориентирано дърво без етикети по върховете с N върха.

За целта трябва да напишете две функции: енкодер и декодер. Енкодерът получава двоичния низ и трябва да построи дърво с върхове с етикети целите числа от 0 до $N - 1$. Комуникационната среда обаче не е надеждна и **"поврежда" дървото** като добавя един допълнителен връх, който е свързан с някой от оригиналните N върха. Декодерът получава само тази "повредена" версия на дървото, която има $N + 1$ върха и N ребра. Освен това ребрата и етикетите на върховете (и върховете на всяко ребро) биват разбъркани на случаен принцип преди да бъдат подадени на декодера. Това симулира подаване на дърво **без етикети** на декодера. Декодерът трябва да разчита единствено на структурата на дървото и не може да предполага нищо за етикетите.

Декодерът трябва да върне като резултат низ с възможно най-дълъг общ префикс с оригиналния низ S **без значение къде дървото е било "повредено"**. Резултатът Ви ще зависи от най-малката дължина на общ префикс сред *всички възможни сценарии за "повреда"* и сред всички подтестове на тестовите.

Подробности по имплементацията

Трябва да имплементирате следните функции:

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

Тази функция получава: цяло число N (броя върхове, които може да ползвате) и булев вектор $data$, представящ двоичния низ S . Трябва да върне като резултат списък от $N - 1$ ребра, образуващи дърво с N върха (0-индексирано).

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

Тази функция получава: цяло число N (оригиналната стойност, подадена на `encode`) и вектор от N ребра, образуващи дърво с $N + 1$ върха ("повредената" версия). Трябва да върне като резултат редица от битове, която трябва да съвпада с оригиналната $data$ във възможно най-дълъг префикс.

За даден тест, грейдърът ще стартира 1 инстанция на енкодера и N различни инстанции на декодера. Във всяка от тези инстанции съответната ѝ функция ще бъде извикана T пъти – по веднъж за всеки подтест на теста.

Вашата програма не трябва да имплементира `main` функция и не трябва да взаимодейства със стандартните вход и изход. Тя ще бъде компилирана заедно със системен грейдър, който ще се грижи за това.

Ограничения

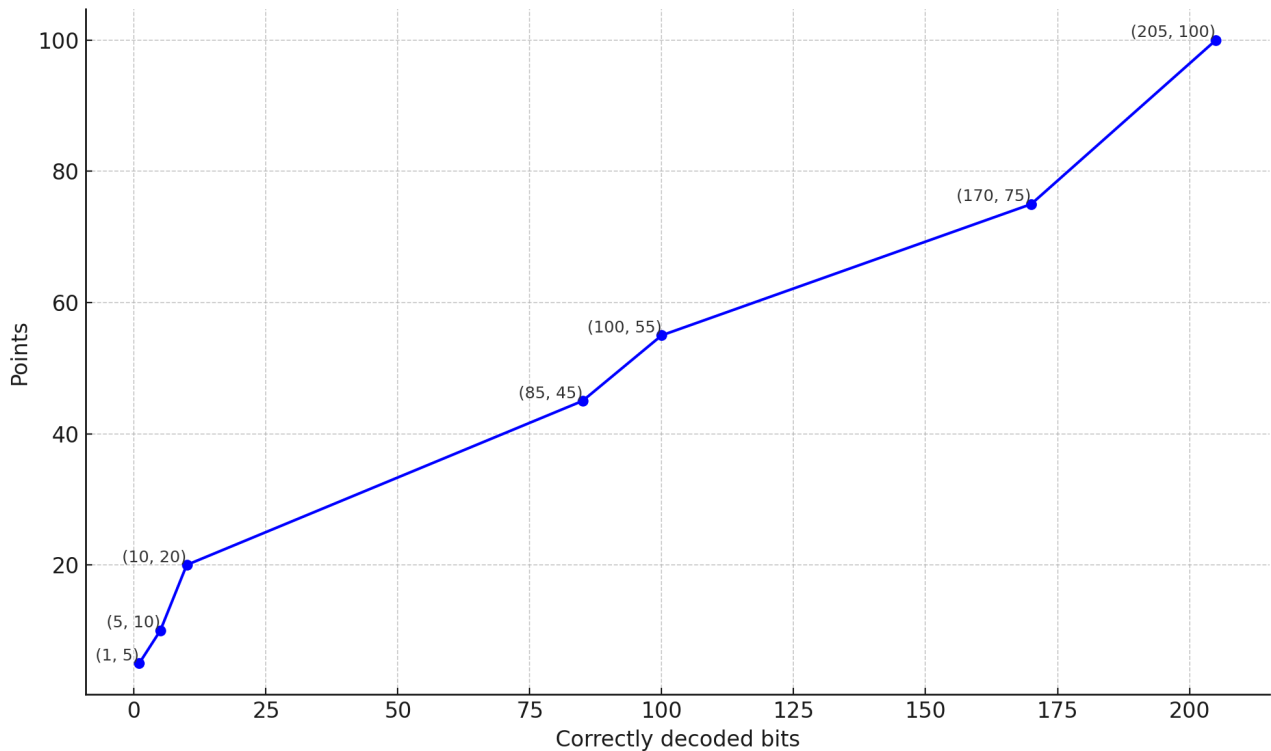
- $T = 2$
- $N = 200$
- $|data| = 2N$
- Етикетите на върховете на "неповреденото" дърво са в интервала от 0 до $N - 1$.
- Етикетите на върховете на "повреденото" дърво са в интервала от 0 до N .

Оценяване

Крайният Ви резултат за задачата (като част от пълния брой точки) се определя от броя коректно декодирани битове от решението Ви.

Ако решението Ви построи невалидно дърво, резултатът Ви ще е 0.

В противен случай, резултатът Ви S се изчислява, използвайки следната пиесно-линейна функция:



Броят коректно декодирани битове е най-малкият сред всички тестове, подтестове и сценарии за "повреда".

Локално тестване

За да тествате решението си локално, е предоставен локален грейдър. Той трябва да бъде компилиран заедно с програмата Ви. Той не използва отделни процеси.

Локалният грейдър симулира пълната процедура на кодиране/декодирание:

1. Прочита цяло число T – броят подтестове.
2. За всеки тест:
 - (a) Прочита цяло число N – броят налични върхове за кодирането.
 - (b) Прочита цяло число ℓ – дължината на двоичния низ (обикновено $2N$).
 - (c) Прочита ℓ бита (всеки от които 0 или 1, без интервали), представящи двоичния низ S .
 - (d) Извиква Вашата функция `encode(n, data)`.
 - (e) Опитва се да "повреди" резултатното дърво като:
 - Итерираща през всяко x — индекс на връх (от 0 до $N - 1$), с който да се свърже новият връх.
 - Добавя нов връх и го свързва с връх x .
 - На случаен принцип разбърква етикетите на върховете, списъците от ребрата и разменя двата края на всяко ребро.
 - (f) Извиква Вашата функция `decode(n, corrupted_edges)` за всяко "повредено" дърво.
 - (g) Сравнява всеки резултат с оригиналния низ и брои колко бита съвпадат с оригиналния низ в най-краткия съвпадащ префикс.

Примерно взаимодействие

Вход	Взаимодействие
1 5 10 1101010110	<code>encode(5, {1,1,0,1,0,1,0,1,1,0})</code> <code>returns {{0,1},{1,2},{0,3},{1,4}}</code> <code>decode(5, {{1,0},{0,2},{2,3},{3,4},{5,3}})</code> <code>returns {1,1,0}</code> <code>decode(5, {{1,0},{0,2},{0,3},{4,0},{5,4}})</code> <code>returns {1,1,1,0,1,0,1,0}</code> <code>decode(5, {{2,0},{2,1},{1,3},{4,1},{5,4}})</code> <code>returns {1,1,1,0,0,0,0,0}</code> <code>decode(5, {{0,1},{0,2},{0,3},{4,3},{3,5}})</code> <code>returns {1,1,1,0,1}</code> <code>decode(5, {{1,0},{1,2},{2,3},{4,2},{5,4}})</code> <code>returns {1,1,1,0}</code>