# UNIVERSIDADE FEDERAL DO PARÁ

DANIEL NAIFF DA COSTA
EDUARDO FERNANDES ALBUQUERQUE

2025

# 1 Jesse and Cookies

A conta usada para enviar esse problema foi a "danielnaiff344". O problema dos cookies trata-se de um desafio onde a doçura dos cookies deve ser aumentada até que todos os cookies possuam uma doçura maior ou igual a um valor de limiar k. A cada operação, são retirados os dois cookies menos doces e combinados em um novo cookie, cuja doçura é dada pela fórmula:

$$\text{nova doçura} = \text{menor doçura} + 2 \times \text{segunda menor doçura}$$

Esse processo de combinação de cookies é repetido até que todos os cookies atendam à condição de doçura ou até que não seja mais possível realizar combinações. O objetivo é determinar o número mínimo de operações necessárias ou indicar que não é possível alcançar o valor desejado de doçura.

2. Implementação

Para resolver o problema de forma eficiente, utilizamos a Heap Mínima. A Heap Mínima é uma estrutura de dados especializada que permite acessar o menor elemento de uma coleção de dados de maneira rápida, o que é ideal para este problema onde precisamos acessar os dois cookies menos doces repetidamente.

Passos do algoritmo:

1. Inserção na Heap: Todos os valores de doçura dos cookies são inseridos na Heap Mínima. A estrutura é ajustada automaticamente para manter o menor valor no topo.

2. Operação de Combinação: Enquanto houver mais de um cookie e o valor mínimo da Heap for menor que k, os dois menores cookies são removidos, combinados e o novo cookie gerado é inserido de volta na Heap. A operação de combinação consiste na fórmula mencionada, onde a doçura do novo cookie é a soma do menor valor com duas vezes o segundo menor valor.

3. Verificação da Condição: A cada operação de combinação, verificamos se o valor mínimo da Heap é maior ou igual a k. Se for, o processo é interrompido e retornamos o número de operações realizadas. Se não for possível atingir o valor k devido à falta de cookies suficientes para combinar, retornamos -1.

Explicação do Algoritmo:

- HeapMinima: A classe HeapMinima implementa os métodos necessários para inserir e remover elementos da heap, além dos métodos auxiliares ajustar cima e ajustar baixo, que são responsáveis por manter a estrutura da heap após cada operação.

- Função cookies: A função começa inserindo todos os cookies na heap e depois verifica se já é possível alcançar a doçura k. Se não for possível, ela combina os dois menores cookies, conta a operação e repete o processo até que a condição seja atendida ou seja impossível continuar.

Exemplo:

- Entrada:

6 7 1 2 3 9 10 12

- Saída:

2

No exemplo acima, as operações realizadas são:

1. Combinação de 1 e 2 para formar 5, resultando em: [3, 5, 9, 10, 12]. 2. Combinação de 3 e 5 para formar 13, resultando em: [9, 10, 12, 13].

Com isso, são necessárias 2 operações para atingir a condição de doçura mínima de 7.

3. Conclusão

A solução foi eficiente utilizando a Heap Mínima, que garantiu a extração rápida dos menores valores. A principal dificuldade foi a manipulação da estrutura para garantir que os cookies fossem combinados corretamente e que a condição de doçura fosse atendida em um número mínimo de operações. O algoritmo apresentou bom desempenho para entradas de tamanho razoável, no entanto, na primeira tentativa de envio apenas um teste não passou, fazendo com que fosse necessário melhorar a complexidade.

# 2 No Prefix Set

1. Introdução

A conta usada para enviar esse problema foi a "danielnaiff344". O problema do prefixo exige que, dado um conjunto de palavras, aponte quais delas é prefixo de outra. Se isso ocorrer, o conjunto de palavras é classificado como um "BAD SET". Caso contrário, é classificado como um "GOOD SET". Para resolver o problema o professor sugeriu estrutura de dados de árvore Trie, que é ideal para armazenar e verificar rapidamente os prefixos de palavras.

2. Implementação

A Árvore Trie é uma estrutura de dados que armazena palavras de tal forma que permite a busca eficiente por prefixos. Cada nó da Trie representa uma letra, e as palavras são formadas pela sequência de nós de um caminho da raiz até as folhas.

Passos do algoritmo:

1. Inserção na Trie: Para cada palavra, verificamos se ela é um prefixo de uma palavra já inserida ou se outra palavra já inserida é um prefixo da palavra atual.

2. Verificação de Prefixo: Durante a inserção, se encontramos uma palavra que já é prefixo de outra, ou se a palavra a ser inserida é prefixada por outra, o conjunto é considerado um "BAD SET".

3. Resultado: Se nenhuma violação de prefixo for encontrada, o conjunto é classificado como um "GOOD SET".

Explicação do Algoritmo:

- TrieNode: A classe TrieNode representa um nó da Trie. Cada nó possui um mapa de filhos e uma flag endOfWord que indica se o nó marca o final de uma palavra.

- Trie: A classe Trie implementa os métodos de inserção e verificação. O método insert tenta inserir uma palavra na Trie e verifica se ela viola a condição de prefixo.

- Função insert: primeiramente inicializa o Nó atual, na sequencia é a feita a iteração sobre os caracteres, através de um for, em seguida é feita a verificação da palavra existente e depois é feita a inserção de novos Nós, após o loop o algoritmo

marca o Nó final da palavra e por último é feito de novo a verificação dos filhos após a inserção.

- Função noPrefix: A função percorre as palavras e insere cada uma na Trie. Se alguma palavra for um prefixo de outra ou for prefixada por outra, imprime "BAD SET" e a palavra problemática. Caso contrário, imprime "GOOD SET".

Exemplo:

- Entrada:

7 aab defgab abcde aabcde bbbbbbbbbb jabjjjad

- Saída:

BAD SET aabcde

No exemplo, a palavra aab é prefixo de aabcde, o que torna o conjunto "BAD SET". A verificação falha na palavra aabcde.

3. Conclusão

A solução foi eficiente utilizando a Árvore Trie, que proporcionou uma forma rápida de verificar a condição de prefixo entre as palavras. A implementação da Trie foi crucial para garantir que as verificações fossem feitas de maneira otimizada, tornando o código capaz de lidar com grandes conjuntos de palavras de forma eficiente. O principal problema enfrentado foi que nem todas as soluções foram aceitas de primeira, exigindo que fosse feita mais verificações através de Ifs na função "insert".