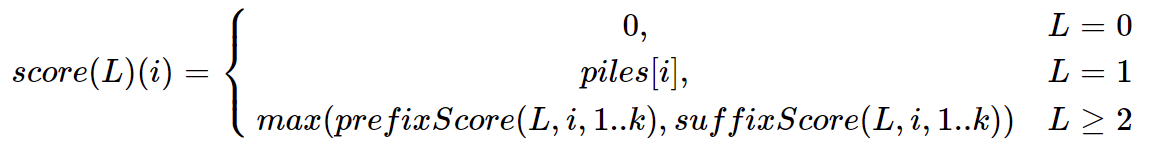
**GameOfBeansII**

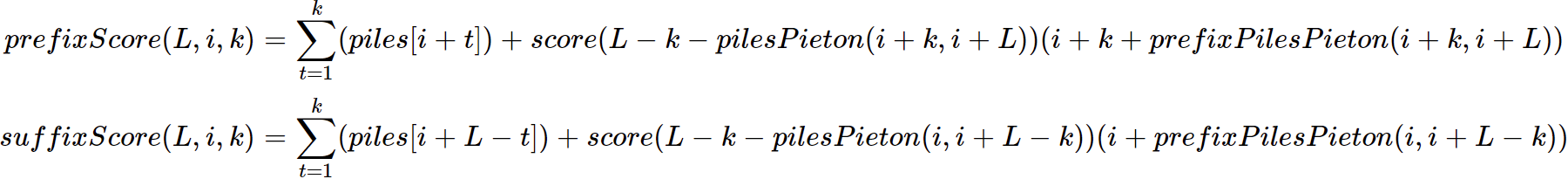
André Santos, nº55415

Tiago Teles, nº54953

**Resolução do Problema:**

score(L)(i) devolve a pontuação máxima que a Jaba consegue ganhar para uma subsequência de pilhas contínua de comprimento L a começar no índice i, quando é a primeira a jogar. Caso seja o Pieton o primeiro a jogar o problema é reduzido a um tratável pela função score.



Onde: 

Em que:

k é o número de pilhas removidas pela Jaba.

pilesPieton(i, j) devolve o número de pilhas removidas pelo Pieton para uma subsequência de pilhas contínua a começar no índice i e a terminar no índice j.

prefixPilesPieton(i, j) é análoga mas contabiliza somente as pilhas removidas do prefixo da subsequência.

Em mais detalhe: o caso base em que L=0 justifica-se pela razão de que para um qualquer subjogo de comprimento 0 é impossível remover pilhas, e sendo a pontuação base 0, esta assim se manterá. Para o segundo caso base (L=1), a Jaba é forçada a jogar pois só há uma pilha a retirar, ficando então com a pontuação dela proveniente. No caso geral, a Jaba considera todas as jogadas que lhe são possíveis, escolhendo de seguida aquela cuja soma da pontuação que acabou de obter com a pontuação que obterá no subjogo que resta após a jogada do Pieton seja máxima.

**Complexidade Temporal**

Caso Pieton seja o primeiro:

- Jogada Pieton: Θ(D)

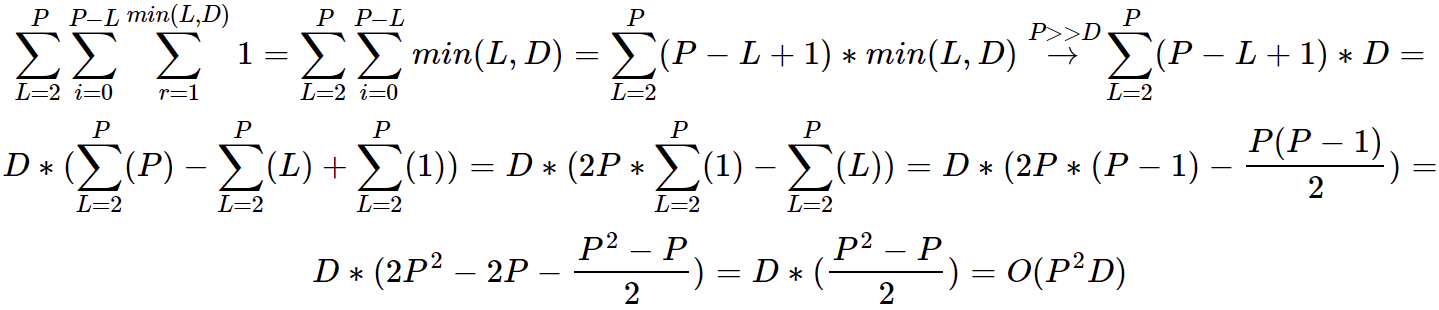
- Reatribuir piles: O(P)

Criar caches Pieton (criar duas matrizes com P\*D de dimensão (ou (P-X)\*D, caso o Pieton jogue primeiro, sendo X o número de pilhas por ele removidas)): Θ(1)

Preencher cache Pieton (preencher todas (com exceção das D últimas linhas, que são preenchidas parcialmente) as posições de memória das tabelas acima mencionadas, em que cada preenchimento tem custo constante): O(PD)

Criar tabela Jaba (criar uma matriz análoga às da cache do Pieton mas invertida): Θ(1)

Preencher tabela Jaba valores iniciais (preencher todas as posições de memória da tabela acima mencionada, em que cada preenchimento tem custo constante): O(PD)

Preencher tabela Jaba: O(P2D) (o erro cometido no cálculo ao assumir P>>D é por excesso. É aceitável pois segundo as restrições do jogo tem-se P≥D, P≤100 e D≤10 pelo que esta é uma aproximação representativa da maioria dos casos. Considerando o caso limite em que P=D e substituindo min(L,D) por L chegar-se-ia a O(P3), que é consistente com O(P2D), dado P=D)

Complexidade total: O(P2D)

Nota: A complexidade é O(P2D) e não Θ(P2D) pois nos casos em que o Pieton joga primeiro as pilhas que o mesmo remove na primeira jogada não são consideradas na iteração das estruturas de dados. Para os restantes casos a situação é análoga.

**Complexidade Espacial**

Jogo corrente: Θ(1)

Cache par (pilhas, pontuação) para jogadas a retirar pilhas do prefixo do Pieton: O(PD)

Cache par (pilhas, pontuação) para jogadas a retirar pilhas do sufixo do Pieton: O(PD)

Tabela função score(L)(i): O(PD)

Complexidade total: O(PD)

Nota: As complexidades são O(PD) e não Θ(PD) somente porque nos casos em que o Pieton joga primeiro as pilhas que o mesmo remove na primeira jogada não são consideradas na criação das estruturas de dados.

**Conclusões**

Em termos de complexidade temporal cremos que a nossa solução não possa ser melhorada pois a componente máxima é a das jogadas da Jaba e advém da definição do problema que a mesma tem de ter conhecimento acerca de todos os subjogos provenientes daquele a ser avaliado (de onde surge a componente P2) e que para cada um destes jogos tem considerar a melhor de 2\*D jogadas possíveis (de onde surge a componente D) pelo que se for necessário consultar cada uma destas alternativas a complexidade temporal terá de ser no mínimo da ordem de grandeza P2D. É necessário consultar todas as alternativas pois as distribuições de pilhas a serem avaliadas pela função score são arbitrárias e as jogadas do Pieton delas dependentes, pelo que não há nenhum padrão geral que permita inferir a partir de um subjogo informação acerca dos restantes.

Em termos de complexidade espacial é de notar que é o uso de uma “matriz circular” que nos permite atingir O(PD). Esta possibilidade deve-se ao facto de no caso geral da função score ser apenas necessário ter acesso aos valores até 2\*D linhas antes.

Tentou-se ainda recorrer a estruturas que armazenassem somente as jogadas do Pieton para um dado subjogo. Estas tentativas, porém, revelaram-se infrutíferas dado que a implementação ficava consideravelmente mais complicada e as complexidades esperadas eram quadráticas.

A solução final por nós adotada será provavelmente uma das melhores, dado que o preenchimento de cada posição das caches tem custo constante e o conjunto (prefixo, profundidade (D), pontuação, jogada) armazenado em cada entrada contém informação generalizável para qualquer variante do jogo em que o prefixo seja igual e o comprimento (P) seja maior ou igual à profundidade (D) (esta explicação é só válida para a cache de jogadas a retirar do prefixo mas no que toca à do sufixo a situação é análoga). Há entradas com valores duplicados, mas não vemos formas eficientes de as evitar.

Já em termos absolutos espaciais seria possível armazenar a tabela da Jaba numa “matriz circular” em que cada linha tem dimensão inferior à anterior em uma unidade, o que se traduziria na poupança de cerca de D2/2 posições de memória. Apesar de ser fácil de implementar, por não ser uma poupança muito significativa e em prole da simplicidade, o anteriormente referido não acabou por se concretizar. Nas caches do Pieton seria possível aplicar algo semelhante para as últimas D linhas das matrizes.

Algo mais interessante na vertente temporal, ainda em termos absolutos, seria guardar a tabela da Jaba com a dimensão resultante do ajuste acima proposto num vetor unidimensional circular, recorrendo a uma função para converter os índices, o que melhoraria do uso da cache do processador, especialmente para valores de P altos durante as últimas iterações de preenchimento.







