Trabalho 2 - Salve as pizzas do IC

MC458
Pedro Carvalho Cintra
247315

---

1. O problema

Ocorrerá um evento no IC onde diversas pizzas serão servidas para todos os alunos da universidade, porém um acidente ocorreu e só temos uma fornalha para assar N pizzas em apenas T minutos. Além disso, como cada ingrediente é bem diferente um do outro, cada pizza i possui um sabor inicial si, um tempo de preparo ti e uma taxa de decaimento do sabor ri. Logo, para uma pizza qualquer sua contribuição para o sabor final total de todas as pizzas é $si \ - \ ri(t \ + \ ti)$, onde t é o tempo esperado pela pizza para entrar na fornalha.

2. A solução

A partir dessas informações que serão fornecidas como entrada, devemos desenvolver um algoritmo que escolha a melhor sequência de pizzas possível que fornece o valor máximo de sabor final total das pizzas.

Ao analisar o problema e construir alguns casos testes para observar como a solução é alterada de acordo com algumas relações feitas entre as propriedades das pizzas ($si, ti\, ri$), percebemos que existe uma preferência entre escolher pizzas que possuem uma valor maior para a relação ri/ti. Além disso, para esse problema, podemos fazer uma analogia com o Problema da Mochila Binária já que uma pizza i, que pode ou não estar na solução, possui um "peso" (tempo que a pizza demora para ser assada) e uma "capacidade"(T) que deve ser preenchida com esse parâmetro.

Por esses motivos e pelo fato que a ordem com que essas pizzas entram na solução influenciam no resultado final, a estratégia escolhida para solucionar esse problema é usar uma Escolha Gulosa para selecionar a melhor ordem de entrada das pizzas e utilizar um algoritmo de Programação Dinâmica para decidir se dada pizza entra ou não na solução.

2.1. A Escolha Gulosa

A Escolha Gulosa nesse caso baseia-se em escolher sobre qual parâmetro ou relação o conjunto de pizzas será ordenado para que o algoritmo produza a solução ótima. Sabendo disso provaremos que a melhor escolha para uma possível pizza i entrar na solução é escolhendo a de maior relação ri/ti.

- *Prova*:

  Dado duas pizzas quaisquer i e j; supondo que já possuímos um sabor total s', para as instâncias já vistas, e que gastamos t' até agora; ao escolher uma possível nova pizza para ser assada, observamos o sabor fornecido ao adicionar primeiro a pizza i e depois a pizza j e vice-versa:

$$s' \ + \ si \ - ri * (t' + ti) \ + sj - rj * (t' + ti + tj) \qquad (i)$$

$s' + sj - rj * (t' + tj) + si - ri * (t' + tj + ti) \ (j)$

Fazendo $i - j$:

$- rj * ti + ri * tj$

Analisando essa equação, vemos que caso ela possua um valor maior que zero, a pizza i deverá ser colocada antes da pizza j, pois significa que o sabor fornecido pela pizza i é maior que o sabor fornecido pela pizza j:

$- rj * ti + ri * tj > 0$

$ri * tj > rj * ti$

$ri/ti > rj/tj$

Portanto, a pizza a ser selecionada para que haja maior fornecimento de sabor será sempre a de maior ri/ti.

2.2. Projeto por Indução do algoritmo

Indução sobre a quantidade N de pizzas:

- *Base*:

  Se $N = 0$, não há nenhuma pizza. $Sabor\ total = 0$.

  Se $N = 1$, caso seja possível assar a pizza no tempo T $Sabor\ total = si$; caso contrário $Sabor\ total = 0$
- *Hipótese de Indução (h.i)*:

  Suponha que para todo i, $1 < i < N$, sabemos resolver o problema de maneira ótima para o tempo T.
- *Passo Indutivo*:

  Tome uma instância do problema com N pizzas e T minutos. Considere a n-ésima pizza desse problema. Se ela não pode ser assada pois $tn > T$, então ela não pode fazer parte da solução, logo a solução para obter o valor máximo de sabor deve considerar apenas a instância do problema com $N - 1$ pizzas e T minutos (conseguimos resolver por *h.i*). Porém, se é possível que ela seja assada, para que a solução seja ótima, o sabor máximo deve ser o valor máximo entre não usar a n-ésima pizza, recaindo para uma instância com $N - 1$ pizzas e T minutos, e usar a n-ésima pizza, recaindo para um a instância com $N - 1$ pizzas e $T - tn$ minutos (em ambos os casos resolvemos por *h.i*).

  Assim, sendo $z[i][j]$ o sabor total máximo para um instância com i pizzas e j minutos, temos a seguinte recorrência:

$$z[0][j] = z[i][0] = 0$$
$$z[i][j] = max\{z[i - 1][j], z[i - 1][j - ti]\}$$

Esse processo só é possível pela presença de uma Subestrutura Ótima. A implementação do algoritmo foi feita de maneira *bottom-up*, ou seja, completando a tabela $N\ x\ T$ das instâncias menores para as maiores.

2.3. A Subestrutura Ótima

É fácil perceber que a solução do problema em questão possui subestruturas em suas instâncias. Porém, para que seja possível realizar o processo indutivo descrito anteriormente, devemos provar que essa subestrutura deve ser ótima para que a solução seja ótima.

- *Prova*:

Dado uma solução ótima S para uma instância de N pizzas e T minutos, existem duas possibilidades: (1) a n-ésima pizza não pertence a S, nesse caso S também é uma solução ótima para a instância com $N - 1$ pizzas e T minutos; para mostrar que ela é ótima para essa instância, vamos supor por contradição que existe S' tal que seu sabor total é maior que o da solução S, para essa mesma instância; porém S' também é solução para a instância de N pizzas e T minutos e sabor total de S' é maior que sabor total de S, o que não pode ser verdade já que S era ótima, para essa instância. (2) a n-ésima pizza pertence a S , vamos considerar $S'' = S \setminus \{n\}$; S'' é solução ótima para a instância de $N - 1$ pizzas e $T - tn$ minutos; vamo supor por contradição que existe S' que é melhor que S'', para essa instância; é perceptível que $S' \, U \, \{n\}$ é uma solução para a instância de N pizzas e T minutos e possui valor melhor que S, o que não pode ser verdade já que S era ótima para essa instância.

3. A complexidade

Verificando os passos do algoritmo feito vemos que primeiramente ele utiliza uma rotina de ordenação que leva tempo $O(N * logN)$. Logo após isso ele realiza iterações que percorrem toda a matriz z de tamanho $N \, x \, T$, ou seja, o restante do algoritmo gasta tempo $O(N * T)$.