1 Enunciado

O Exercício a seguir foi extraído do livro Algorithm Design de John Kleinberg e de Eva Tardos, capitulo 4, sendo o exercício de numero 7. Vamos expôr apenas um resumo do enunciado, com as principais informações para resolução de tal exercício. Destaca-se aqui, que a escolha desse exercício foi pautada pelo interesse em enunciar e provar a eficiência de um algoritmo guloso.

Dado um problema, um algoritmo guloso tem como principal característica construir uma solução em pequenos passos, de modo a escolher uma ação a cada passo sob um determinado critério, com o objetivo de otimizar a solução final. Dessa forma, ao final da rotina, deve-se retornar como saída uma solução ótima. É possível identificar diferentes critérios para resolução de um problema, por intermédio de um algoritmo guloso, porém, não necessariamente, o critério escolhido nos leva à solução ótima. Dessa forma, após construir um algoritmo guloso, se faz necessário provar que a solução proposta por tal algoritmo é ótima, sendo esse o nosso objetivo com esse exercício.

O supercomputador exerce trabalhos diferentes que possuem tempo final P1, P2,...,Pn. Um trabalho exercido pelo supercomputador Pi só se inicia quando outro Pi-1 termina. Após o trabalho do supercomputador, é necessário o trabalho de um PC para processamento dos dados. Os PCs possuem tempo final de trabalho da ordem de f1,f2,...,fn. Pergunta-se: Qual a melhor forma de alocar os trabalhos dos supercomputadores e dos PCs de modo a minimizar o tempo de processamento do dado.

2 Prova de Otimalidade

A primeira etapa será analisar os jobs do supercomputador. Como ja dito no enunciado, Pi será o valor da duração dos jobs do supercomputador e Fi o valor duração dos jobs dos PCs.

Uma vez que todos os jobs de um PC iniciam necessariamente após o job de um supercomputador, e como os jobs dos supercomputadores devem ser ordenados de modo que nao haja sobreposição entre seus valores, podemos inferir que a ordem dos supercomputadores é indiferente, pois dadas as hipoteses listadas, o que de fato irá interferir no tempo final dos jobs será a ordem dos trabalhos dos PCs.

Com efeito, nosso algoritmo A exposto em código python na pròxima seção ordena a duração dos jobs feitos pelos PCs em ordem decrescente. Em resumo, nosso critèrio será escolher a cada passo o job de um PC de maior valor. Através desse algoritmo, vamos ser capazes de alocar os jobs dos supercomputadores e dos PCs de modo a gastar o menor tempo possíivel nessa tarefa, como será provado a seguir:

Passo Base: Seja k a iteração no tempo, temos nesse passo k = 1.

Seja Fi o job de maior duração feito por um PC, supõe-se um algoritmo O que não posiciona o job Fi* na primeira posição, isto é, após o termino do job do primeiro supercomputador. Isso nos leva à conclusão de que o algoritmo O*

não ordena os trabalhos dos PCs em ordem decrescente dos valores de duração. Aqui, destaca-se o fato de que para um job de um PC começar o de um supercomputador deve terminar e, no passo base, apenas um job P1 foi concluído por um supercomputador. Seja T o tempo gasto pelo algoritmo A implementado nesse artigo e seja T‘ o tempo gasto pelo algoritmo O, pode-se afirmar que:

$$T = P1 + Fi* < P - 1 + Fi* = T‘ \tag{1}$$

Ao considerar P-1 como P não um, sabemos que tal job começará necessariamente após P1. Ou seja, se Fi for posicionado em outro lugar que não P1 por O, o algoritmo A termina em menor tempo. Por outro lado, se Fi* é posicionado em P1 por O, como em A, temos que os dois seguem o mesmo valor ótimo T. Assim o algoritmo A mantém a propriedade de s̈tays ahead(̈nota de rodape) de algoritmos gulosos.*

Passo Indutivo: Aqui, considera-se que as iterações k =¡ n são válidas.

A partir da hipótese indutiva podemos dizer que:

$$T = Pk + 1 + Fi - k* < Pk + j + Fi - k* = T‘, sendo j > 1. \tag{2}$$

Isso é válido pois Pn+j necessariamente começa após Pn+1 o que faz do tempo em O ser maior do que em A. Se j = 1, então o algoritmo O possui tempo igual ao tempo de A, o que garante a propriedade de stays ahead. Dessa forma, pode-se concluir que nosso algoritmo é ótimo.