Achar um limitante inferior para algoritmos de ordenação por comparação.

- Ou seja, considerando o conjunto de todos os algoritmos de ordenação, é possível demonstrar que eles não podem ter complexidade de tempo menor?
- Por exemplo: “A complexidade de tempo desses algoritmos não pode ser melhor que nlogn”
- Sinônimo: “não existe algoritmo de ordenação por comparação que seja melhor que nlogn.”
- Lembre que heap e merge são teta-grande (nlog(n)), ou seja, o limitante não pode ser pior que nlog(n).
- como demonstrar??

Comparar dois elementos e tomar alguma ação

- É o que um algoritmo de ordenação por comparação faz.
- Devido a estrutura da operação de comparação, há apenas duas ações possíveis (por exemplo trocar os elementos ou não). Olhando para a ramificação (em dois) que tipo de árvore é??
- Podemos representar a comparação entre elementos como um nó de uma árvore. A decisão tomada (trocar ou não) corresponde a passar por uma aresta (e chegar a um outro nó e eventuamente tomar uma outra decisão)
- Até que nenhuma decisão precise ser tomada e a permutação de elementos (implícita no nó) seja a permutação desejada.

Desta forma uma determinada execução do algoritmo corresponde a percorrer um caminho na árvore

- A árvore representa todos os caminhos que o algoritmo pode percorrer. Como apenas as comparações são suficientes para determinar esse caminho, chamamos a esta árvore de árvore de decisão.
- No mínimo, quantas folhas tem a árvore?
- Uma árvore completa tem número máximo de folhas – qual a relação entre número de folhas e altura?
- Qual a menor altura que tal árvore pode ter?
- (note que se algum caminho for mais curto, outro(s) tem que ser mais longos para acomodar as folhas e isto acaba aumentando a altura da árvore)

Isto quer dizer que a árvore mais baixa tem todas as folhas na mesma altura

- … se todas as folhas estão na mesma altura e dado que são ao menos n! A altura tem que ser ao menos...

log(n!)

- Qual o limitante superior para n!??
- Nota: O limitante inferior não acomoda a quantidade necessária de folhas.
- O limitante superior muito folgado levaria (por exemplo) à conclusão de que algoritmos de ordenação tem que ser omega-grande(n^2), o que é uma contradição pois merge e heap são teta-grande (nlogn)

n^n

- É um limitante para n!
- Outro limitante é dado pela aproximação de Stirling:
-
- N!=sqrt(3*PI*n)(n/e)^n(1+Teta(1/n))
-
- Qualquer um deles leve à mesma conclusão:
-

nlog(n)

- É a altura mínima da árvore.
- Ou seja, algoritmos de ordenação, por melhor que sejam, não são mais rápidos que nlog(n).
- Consequentemente merge e heap são algoritmos ótimos do ponto de vista assintótico.
- Embora quicksort não seja assintoticamente ótimo (pois o pior caso é n^2), sabe-se que o caso médio é O(nlog(n)) e a constante que acompanha essa complexidade é menor que a de merge e heap, logo, na comparação direta, quick é mais rápido.