FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Trabalho realizado por:

João Lopes	2016244868
Roman Zhydyk	2016231789
Tiago Pessoa	2016242888

1. Design of the experiment

Problem statement

De que maneira as falhas de memória afetam os algoritmos de ordenamento?

Identify variables

Variáveis dependentes:

- Array original;
- Array ordenado pelo algoritmo escolhido, com probabilidade de falha de memória;
- Array ordenado pelo algoritmo escolhido, sem probabilidade de falha de memória;
- Maior sub sequência detectada no algoritmo de ordenamento.

Variáveis independentes:

- Probabilidade de uma falha de memória ocorre, número real entre 0 e 1;
- Tamanho do array ordenado, com máximo de 10000;
- Tipo de algoritmo (quicksort, mergesort, insertionsort e bubblesort)

Levels:

- Variáveis contínuas:
    - Tamanho dos arrays, probabilidade de falha.

- Variáveis discretas:
    - Tipo de algoritmo.

Generate hypothesis

Antes de realizar as experiências idealizou-se que:

- Aumentando a probabilidade de falha, a maior sub sequência ordenada diminuía de tamanho;

- A partir de certa probabilidade os algoritmos não iam conseguir ordenar;
- Um algoritmo de força bruta como o bubblesort iria falhar mais menos pois está sempre a realizar comparações com valores anteriormente ordenados.

Experimental setup

Foi alterado o script fornecido, gen.py, para de uma vez só gerar todos os inputs e outputs necessários para depois se proceder ao tratamento de dados. Foi utilizada a distribuição uniforme.

Para dos dados produzir gráficos foi realizado um script em Python que ia lendo os outputs e produzindo os gŕaficos.

Foi produzido um README.info com mais informação sobre como operar o setup.

2. Measurements, Analysis and Conclusion

Para haver uma maior variabilidade nos dados e consequentemente maior precisão nos gráficos, possibilitando então tirar melhores conclusões, foram produzidos 10 ficheiros de output para cada probabilidade dentro de cada algoritmo.

A probabilidade varia no intervalo [0.01, 0.91] de 0.1 em 0.1.

Depois da leitura dos ficheiros, são guardados os valores Xi da maior subsequência e calculada a respectiva média Xñ com os 10 valores distintos.

Com este conjunto de dados, temos então para cada algoritmo:

- As respectivas probabilidades
- A média Xñ dos valores Xi, da maior subsequência para cada algoritmo e probabilidade
- Os respectivos valores Xi, usados para calcular a média Xñ
- O desvio padrão para cada valor da média Xñ
- Os tamanhos N dos arrays

A estrutura dos dados fica com este aspeto:

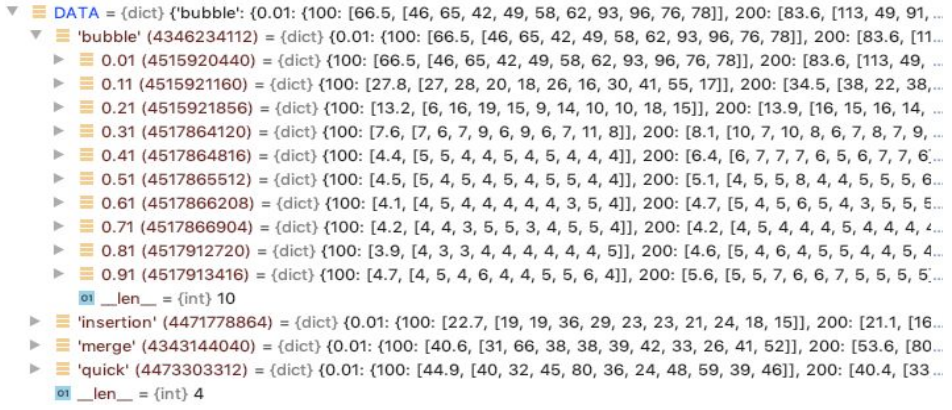


Figura 0.

Com este conjunto de valores, já é possível produzir gráficos relevantes.

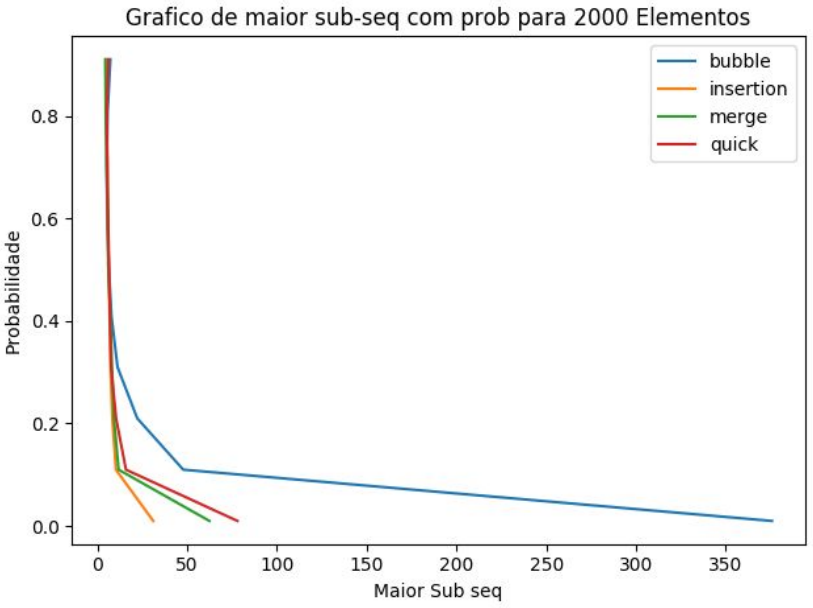


Figura 1.

Pode-se observar que à medida que a probabilidade aumenta, a maior sub sequência ordenada diminui. Pois com maior ocorrência de falhas de memória os algoritmos tendem a ter menos elementos consecutivos ordenados.

A partir de probabilidade 0.5, todos os algoritmos se tornaram iguais, isto é, a maior sub sequência tende para valores pequenos, < 10.

Praticamente não há ordenamento a partir de probabilidade 0.5 até 1.

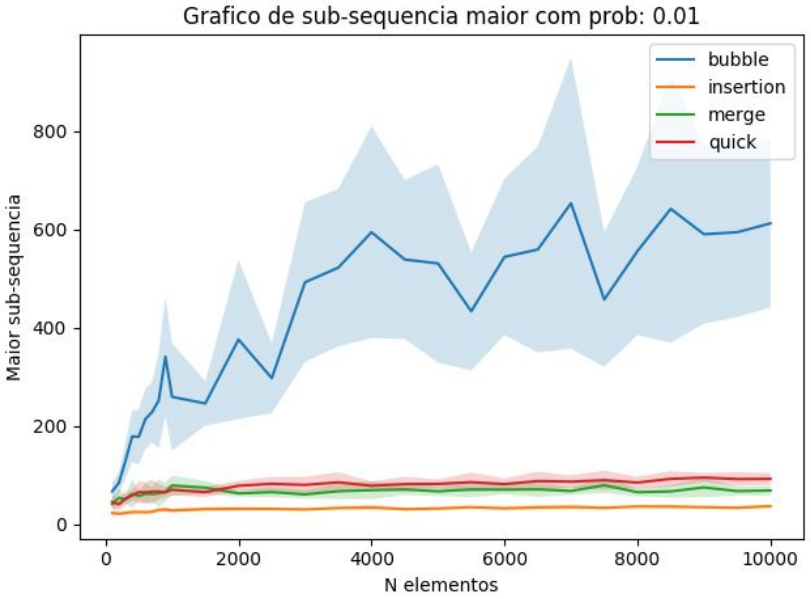
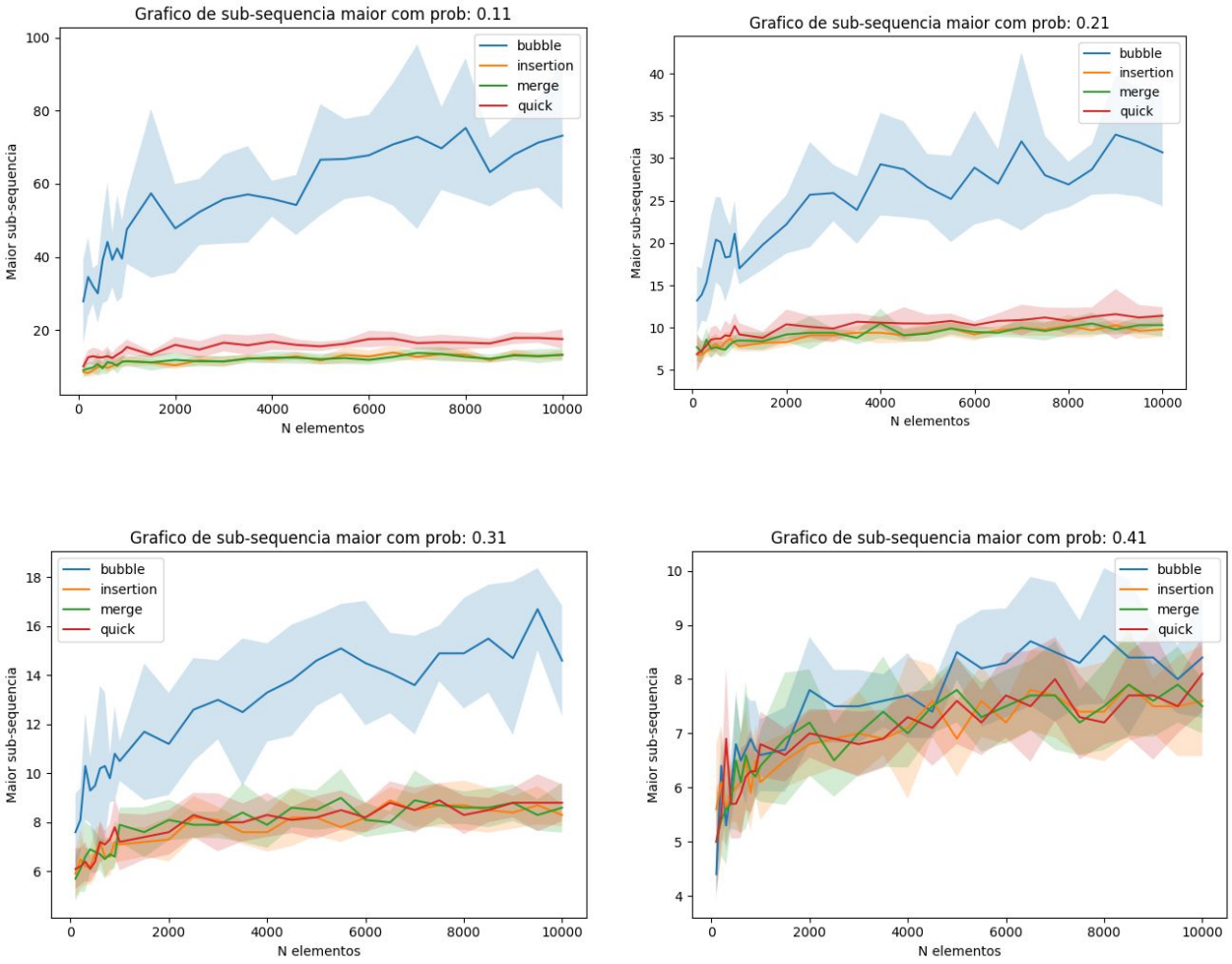


Figura 2.

O bubblesort consegue sequências maiores ordenadas devido a sua natureza, visto que é um algoritmo que percorre sempre o array todo, a cada iteração, e comparando dois números consecutivos verifica se existe algum número por ordenar, realizando trocas para o levar para a sua posição ordenada. Nos outros algoritmos não é realizada esta verificação a cada iteração, são otimizados e têm complexidade temporal menores, o que faz com que se interrompam sequências devido às falhas.

Para probabilidade 0.01 de falha de memória, o bubble apresenta um desvio padrão consideravelmente elevado em relação aos outros algoritmos de ordenamento. Isto deve-se ao facto do input ser uma sequência random e o desempenho deste algoritmo depender sobretudo disso, isto é, se a sequência estiver mais desordenada o algoritmo vai ter realizar mais acessos à memória (mais trocas). Assim, posto isto aliado ao facto de ter subsequências grandes comparativamente aos outros, concluímos que possam existir desvios em relação ao padrão, sempre que se testa um input diferente.



Figuras 3, 4, 5 ,6

Pela análise dos gráficos podemos retirar que à medida que a probabilidade de erro vai aumentando a subsequência maior também diminui como era de esperar, pois a probabilidade de ocorrer erro é maior (isto acontece para todos os algoritmos).

Para a mesma probabilidade o bubble sort apresenta uma subsequência maior isto deve-se ao facto do que já foi referido acima na figura 2.

Para a probabilidade de 0.41 não podemos tirar muitas conclusões porque o valor é muito próximo de 0.51 e como vamos referir mais tarde os valores, à medida que nos aproximamos da probabilidade de 0.5 começam a ser aleatórios pois a probabilidade de ocorrer erro é igual à de não ocorrer.

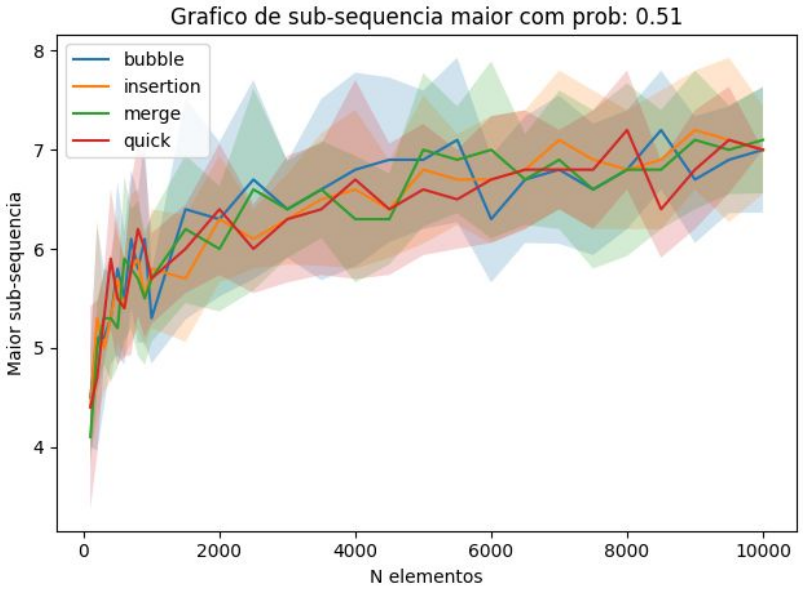
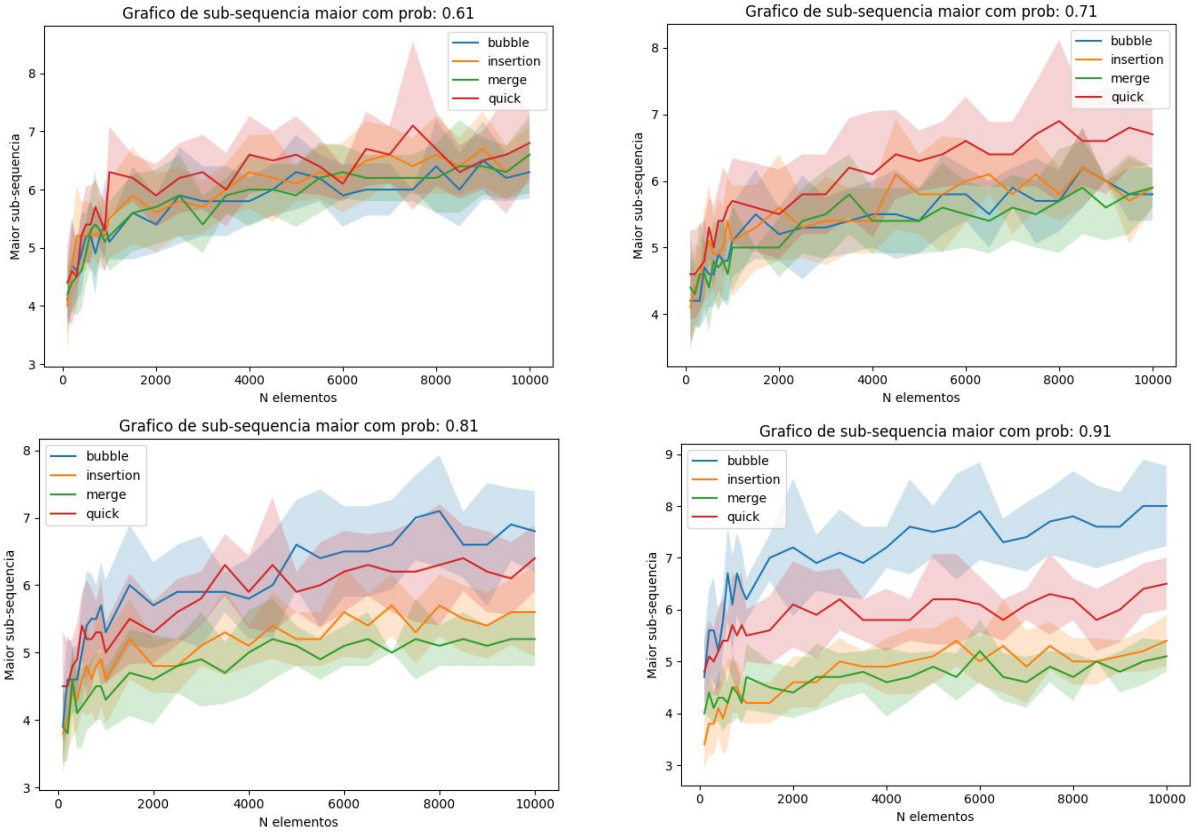


Figura 7.

Para a probabilidade de 51% podemos reparar que os valores da maior subsequência não tendem a variar muito com o aumento do número de elementos do array,e a pequena variação que existe deve-se ao facto do tamanho do array ser maior e consequentemente existir uma maior probabilidade de estarem mais elementos ordenados mas não por consequência dos algoritmos de ordenamento, mas sim pelo facto de haver uma aleatoriedade de estarem alguns elementos consecutivos.

Este padrão acontece cada vez mais quando os valores da probabilidade de ocorrência de erro se aproximam de 0.5.

Este padrão também acontece para valores superiores a 0.5 também:



Figuras 8, 9, 10, 11

Olhando para estes gráficos, pode-se pensar que os valores entre os algoritmos começam a ganhar um certo espaçamento, mas isso acontece simplesmente de forma aleatória, basta olhar para a escala dos Y's e reparar que a variabilidade é muito pequena.

Durante a realização dos testes também reparamos que quanto mais a probabilidade se aproxima de 1, maior a tendência do array ficar ordenado de forma decrescente, isto deve-se ao facto do algoritmo entrar mais vezes na opção de erro e trocar os números do array de forma inversa.

Se a probabilidade de erro for 1 todas as comparações vão ser inversas e o array vai ficar todo trocado ou seja ordenado de forma decrescente.

Para a probabilidade de 1 a subsequência maior será igual a maior sequência de números repetidos, por este motivo não vimos interesse em representar o caso de probabilidade igual a 1.

Concluindo, pela análise realizada o algoritmo de força bruta e de complexidade temporal maior, bubblesort, consegue obter o maior tamanho de subsequência ordenada comparativamente aos outros algoritmos caso hajam falhas de memória, com probabilidade menor que 0.5.