Algoritmos e Estruturas de Dados III
Documentação Trabalho Prático 3 - Legado da Copa

Aluno: Semar Augusto da Cunha Mello Martins
Metricula: 2016006905

Introdução:

A rua Pai no Bar é uma das ruas mais movimentadas em épocas comemorativas. Nela existem somente casas e bares, sendo que os donos dos bares, por serem muito desconfiados, moram na rua da frente do seu bar, pois assim conseguem ver se há alguma movimentação estranha no mesmo. Com a aproximação da copa da Russia, as pessoas da rua querem manter a tradição de dependurar bandeirolas enfeitando a rua. Contudo, devido à crise, não foi possível comprar novas bandeirolas, e, por isso, serão usadas as que os donos dos bares usam em seus estabelecimentos, mas com algumas condições:

- Cada bar contribui com uma linha de bandeirola.
- As bandeirolas serão penduradas com uma ponta no bar e outra na casa do dono do bar.
- As linhas das bandeirolas não podem se cruzar.

Cada bar não fica necessariamente na frente da casa do dono. Logo, foi montada uma tabela onde a primeira coluna indica o numero do bar e a outra o número da casa do dono. Sabe-se que um lado da rua tem números ímpares e o outro números pares.

Modelando o Problema:

É possível modelar esse problema da seguinte maneira:

- Gravamos a tabela na memória em dois vetores, “left” e "right".
- Colocamos os números ímpares em um vetor e os pares em outro.
- Ordenamos um dos vetores de forma crescente, mantendo as duplas nas posições corretas. ou seja, considerando o seguinte cenário: (2,7),(8,5),(4,9), ao se ordenar o vetor "left", o par ordenado inteiro se move, ficando assim: (2,7), (4,9), (8,5). Note que o vetor “right" continuará desordenado. Com essa configuração, é fácil perceber que, se o vetor right na posição n+1 for menor do que o da posição n, então as linhas estão cruzando. Ou seja, estamos procurando pela maior subsequência crescente do vetor “right”.

Metodologia:

A solução do problema foi feita de 3 formas:

1 - **Força Bruta**: A solução por força bruta foi feita com o uso de um vetor binário tão grande quantas bandeiras existirem e um contador binário, se o vetor binário na posição n for 1, então comparamos com o proximo 1 que existir no vetor binário e, caso right[n+x] < right [n], então há um cruzamento na combinação atual. Dessa maneira, o programa é tenta toda e qualquer possibilidade de combinação de bandeiras até achar uma combinação ótima.

2 - **Guloso:** O algoritmo guloso usado foi uma modificação do Patience Sorting para que ele apenas contasse qual seria a maior subsequência crescente. Ele

ontem a solução ótima em tempo n^2. Caso tivesse sido implementada com uma busca binária, seria O(n * log(n)). Ele é um algoritmo guloso pois ele monta as pilhas sempre tentando colocar as “cartas” na pilha mais à esquerda (que em questão de código significa a pilha de índice menor), sendo a condição que determina se é possível ou não colocar a “carta” na pilha é se o valor dela é menor do que o valor da carta que está no topo da pilha atual.

Esse algoritmo obtém a solução ótima do problema de maior subsequência crescente, como é demonstrado matematicamente aqui:
http://statweb.stanford.edu/~cgates/PERSI/papers/Longest.pdf

3 - **Programação Dinâmica:** A solução por programação dinâmica usa um vetor auxiliar que obedece às seguintes regras:

- Se o vetor principal na posição i for o menor valor visto até o momento, atualize o valor do vetor aux na posição 0 (tail[0])
- Se o vetor principal na posição i for o maior valor dentre todas as listas ativas, então o coloque em todas elas.
- Se o vetor principal na posição i não for nem o maior nem o menor, execute uma busca binária e encontre uma lista que possui o maior elemento dela menor do que vetorPrincipal[i] e incremente a lista por esse valor. Então descarte as outras listas de mesmo tamanho.

Exemplo retirado do teste toy_0, depois de ordenar o vetor left:
right = {11, 15, 5, 1, 9, 7, 13, 3}

right[0] = 11 - caso 1, novo menor elemento
11.

right[1] = 15 - caso 2, clonar e aumentar a lista
11, 15

right[2] = 5 - caso 1, novo menor elemento
5, 15

right[3] = 1 - caso 1, novo menor elemento
1, 15

right[4] = 9 - caso 3, montar nova lista e descartar as outras
1, 9

right[5] = 7 - caso 3, montar nova lista e descartar as outras
1, 7

right[6] = 13 - caso 2, clonar e aumentar a lista
1, 7, 13

right[7] = 3 - caso 3, montar nova lista e descartar as outras de mesmo tamanho
1, 3, 13. -> note que essa não é uma lista que possui a maior subsequência crescente, ela serve somente para determinar o tamanho da maior subsequência

crescente. Caso fosse necessário saber qual é a maior subsequência crescente (ou pelo menos saber uma delas, caso haja mais de uma), seria necessário usar um outro vetor que guarda os indexes dos números que determinam a maior subsequência crescente.

Análise de Complexidade:
Dynamic.c - CeilIndex - Espaço - O(1)
Tempo - O(1)

DynamicLIS - Espaço - O(n)
Tempo - O(n * log(n))

bruteForce.c - bruteForceLIS - Espaço - O(n)
- Tempo - O(2^n)

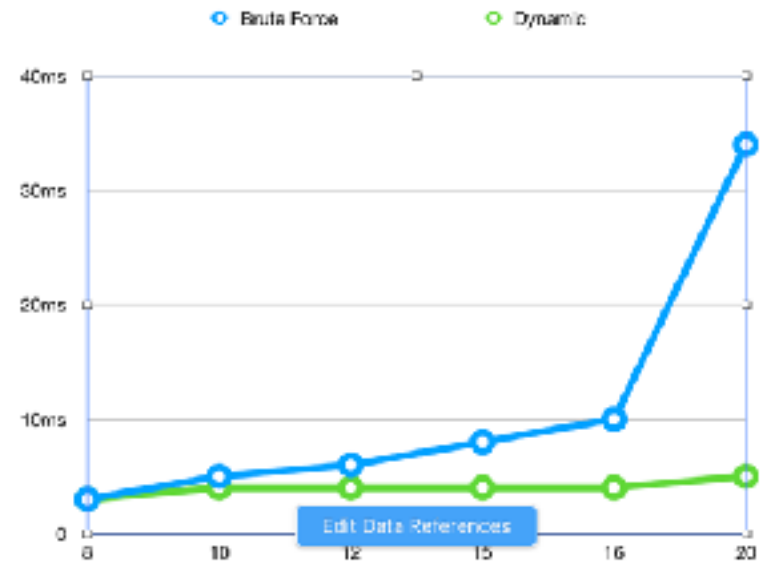
greedy.c - patienceSorting - Espaço - O(n)
- Tempo - O(n^2) - poderia ser melhor (n*log(n)) se tivesse usado uma busca binária para encontrar em qual pilha a carta deveria ser inserida.

quicksort.c - quickSort - Espaço - O(log(n))
- Tempo - O(n * log(n))

main.c - main - Espaço - O(n)
- Tempo - O(tempo da função chamada pelo caso teste) - já que a complexidade da função main é determinada por qual algoritmo foi usado, o dinâmico, o bruto ou o guloso.

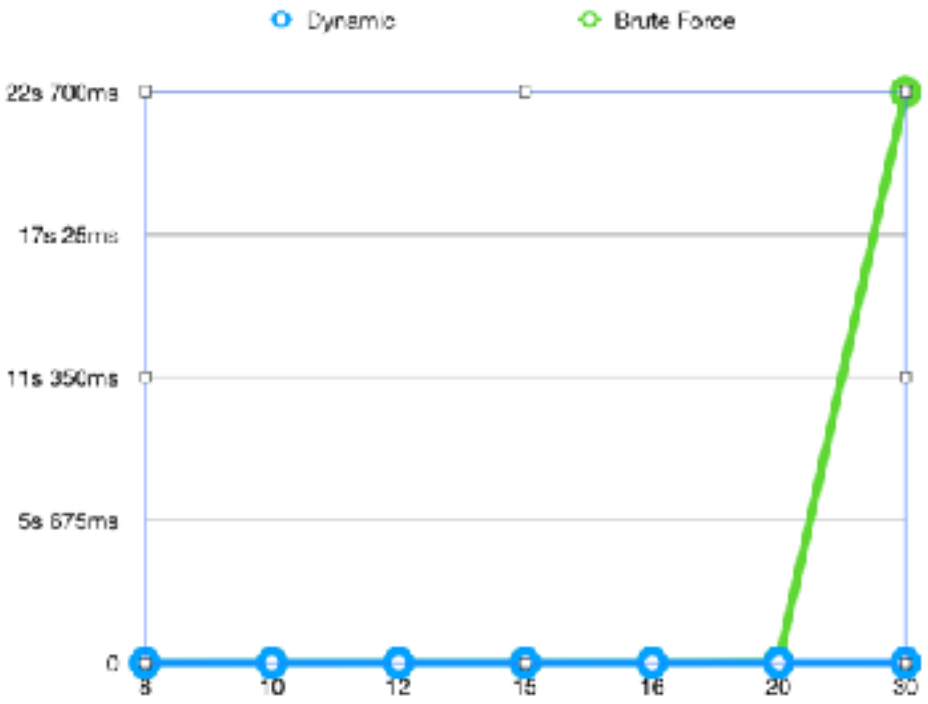
Experimentos:
Os experimentos foram feitos com os testes toys e 9 testes maiores, que variam entre n = 2600 e n = 101447. Os testes foram executados em um macOS Sierra, com 16GB de memória RAM e um intel core i7 com 2.2GHz.

Análise de Resultados:

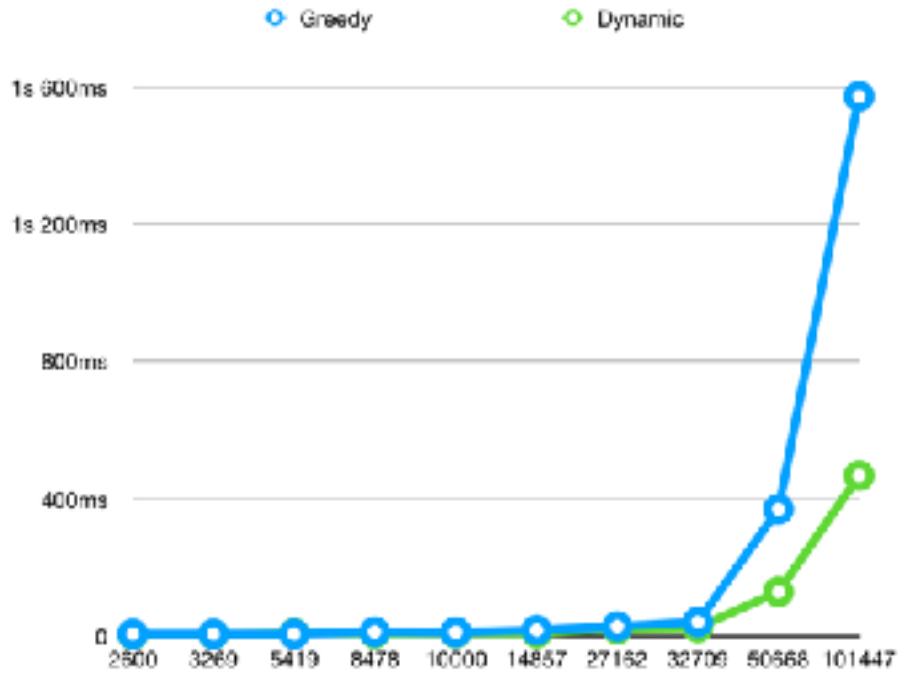


Brute Force x Programação Dinâmica - toy_0 até toy_5

Brute Force x Programação Dinâmica - toy_0 até toy_6



Guloso x Programação Dinâmica - big1 até big9 (n = 2600 até n = 101447):



Conclusão:

Dependendo do tamanho do quarteirão os moradores conseguem determinar quantas bandeirolas serão colocadas somente com um papel e uma caneta. Contudo, à medida de que o número de bares e casas vai aumentando, o uso de um computador e um algoritmo mais eficiente se torna necessário, pois demorará demais para se determinar um número ótimo apenas usando papel e caneta.