Departamento de Ciência de Computadores FCUP

Desenho e Análise de Algoritmos CC2001 2018/2019

Folha 2

1. Pretendemos uma implementação em linguagem C do algoritmo de Graham (*Graham scan*) para cálculo do invólucro convexo de n pontos do plano, descrito nas aulas. Admitir que os pontos têm coordenadas inteiras. **Garantir que a complexidade do algoritmo implementado é** $O(n \log n)$. Assumir que cada ponto tem coordenadas inteiras e é representado por uma estrutura do tipo `PONTO` assim definida:

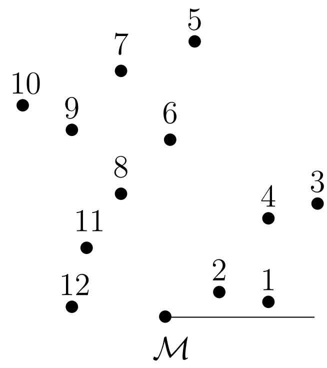
```
typedef struct ponto {
   int x, y;
} PONTO;
```

Definir funções:

a) `int viragem_esq(PONTO p1, PONTO p2,PONTO p3)` para verificar se uma sequência de três pontos (P_1, P_2, P_3) define uma viragem à esquerda, i.e., se a componente não nula do *produto vetorial* do vetor P_1P_2 pelo vetor P_1P_3 (no espaço) tem sinal positivo. Retorna 1 se definir e 0 se não definir.

b) `int ponto_ord_minima(PONTO p[], int n)` para encontrar o ponto $\mathcal{M}$ que tem ordenada mínima (se existirem vários, tomar o de abcissa máxima), sendo dado um *array* com n pontos. Retorna o índice desse ponto no *array* `p`.

c) `void ordena(PONTO p[], int a,int b)` para ordenar o segmento $[a, b]$ do vetor `p` por ordem crescente de ângulo polar crescente relativamente a $p[0]$, que se supõe ter já o ponto $\mathcal{M}$, acima referido (assuma $0 < a \leq b < n$). Deve **adaptar a função mergesort**, descrita nas aulas.
Notar que Q tem ângulo polar maior do que P se $(\mathcal{M}, P, Q)$ constitui uma viragem à esquerda. Se $\mathcal{M}$, P e Q forem colineares (caso em que o produto vetorial é nulo), aparecerá primeiro o ponto que estiver mais afastado de $\mathcal{M}$ (para o determinar, analisar o sinal do *produto interno* dos vetores $\mathcal{M}P$ e PQ).



d) `int convexhull_Graham(PONTO p[], int n)` que determina em `p` a sequência de vértices que define o invólucro convexo dos pontos dados em `p`. Retorna ainda o número de pontos nessa sequência. Assumir que os n pontos estão em posição geral, isto é, que não existem três (ou mais) pontos colineares.

2. Seja S um conjunto ordenado de inteiros não negativos inferiores a N (dado) e seja $\bigcup_{i=1}^{n}[a_i, b_i]$ com $0 \leq n < N$ uma representação canónica de S como união de intervalos, com $a_i \leq b_i$ e $1 + b_i < a_{i+1}$, para todo i. Se $n = 0$, o conjunto S é vazio.

Para cada alínea, escreva a função pedida em pseudocódigo, traduza-a para linguagem C e caracterize a sua complexidade temporal assintótica, se S for representado por:

- **(caso A)** um vetor com N posições, em que a posição k indica se $k \in S$ ou se $k \notin S$;
- **(caso B)** uma matriz com pelo menos n linhas e duas colunas, que guarda a sequência de intervalos $[a_i, b_i]$, ordenada;
- **(caso C)** uma lista ligada simples, ordenada, em que cada nó tem $[a_i, b_i]$ e o identificador do nó seguinte (a lista será vazia se $S = \emptyset$);

Pretendemos algoritmos **eficientes**, em cada caso. Nas alíneas 1c) e 1d), as funções alteram S e preservam a forma canónica descrita.

a) Determinar o número de elementos de S.

b) Verificar se um inteiro x pertence a S ou não.

c) Determinar $S \setminus \{x\}$, para x dado.

d) Determinar $S \cup [a, b]$ para um intervalo $[a, b]$ dado tal que $S \cap [a, b] = \emptyset$, com $0 \leq a \leq b < N$.

e) Escrever S na saída padrão (*standard output*) agrupando todos os pontos isolados num só conjunto, que será escrito no fim. Para $[3, 10] \cup [15, 15] \cup [25, 25] \cup [34, 36] \cup [50, 50] \cup [125, 127]$, deve escrever `$[3,10] \cup [34,36] \cup [125,127] \cup \{15,25,50\}$`. Analogamente, escrevia `$[3,10] \cup [34,36] \cup [125,127]$` se não tivesse pontos isolados, e `$\{15,25,50\}$` se só tivesse pontos isolados. Se $S = \emptyset$, deve escrever `$\{\}$`.

3. Resolver os problemas "Bacalhaus congelados" e "Construção de mapa". A estrutura de dados que representará o grafo deve ser baseada em listas de adjacências (adaptar a definição que consta do arquivo disponibilizado). Analisar a **complexidade** temporal assintótica dos algoritmos implementados.