Instituto Superior Técnico - TagusPark
Matemática Discreta 2020/2021
Exercícios para as aulas de problemas e teorico-práticas

Lista 5

Após a aula teorico-prática e a de problemas da semana em que a lista foi publicada, os alunos deverão tentar resolver todos os exercícios que não foram resolvidos nas aulas. Se tiverem dificuldades ou dúvidas deverão contactar os docentes da disciplina. Vários dos exercícios (ou muito semelhantes) são apresentados como exemplos ou exercícios resolvidos nos Capítulos 6 e 2 do livro (alguns de entre eles estão explicitamente indicados abaixo).

1 Exemplos da aplicação de somatórios e funções geradoras à análise de algoritmos

No âmbito da *análise de algoritmos* é necessário estimar o número de operações relevantes que são realizadas quando o algoritmo é executado. Estas dependem do tipo de algoritmo: por exemplo, operações aritméticas no caso de algoritmos numéricos (e.g. algoritmo de eliminação de Gauss), e operações de comparação no caso de algoritmos de pesquisa e ordenação (e.g. *MergeSort*). Tais contagens são uma aplicação das noções de *somatório* e *forma fechada de somatório* no caso de *algoritmos imperativos*, e da noção de *função geradora* (designadamente resolução de recorrências através de funções geradoras) no caso de *algoritmos recursivos*. Os exemplos seguintes ilustram estas aplicações em alguns casos simples.

```
from math import factorial

def g(n):
  s=[n]
  p=factorial(n)
  for k in range(2,n):
     s.append(p//factorial(n-k))
  return s
```

Fig 1

```
def h(n):
  s=[]
  for k in range(1,n):
    a=n
    for i in range(1,k):
       a=a*(n-i)
    s.append(a)
  return s
```

Fig 2

```
def GaussElimination(A,b):
   Aux=A
   baux=b
   n=len(b)
   for k in range(n-1):
      for i in range(k+1,n):
         m=Aux[i][k]/Aux[k][k]
         Aux[i][k]=0
         for j in range(k+1,n+1):
            Aux[i][j]=Aux[i][j]-m*Aux[k][j]
         baux[i]=baux[i]-m*baux[k]
   return [Aux,baux]
```

Fig 3

1. A função *Python* `g` recebe um inteiro `n`$\in \mathbb{N}_2$ e devolve uma lista `s` de comprimento `n-1` na qual o primeiro elemento é o número de arranjos n, 1 a 1, o segundo elemento é o número de arranjos n, 2 a 2, etc. Por exemplo, `g(5)` devolve `[5,20,60,120]`. Suponha-se que ao avaliar `factorial(x)` são executadas $x-1$ multiplicações quando $x \in \mathbb{N}_1$. Conclua que, para cada $n \in \mathbb{N}_3$, o número de divisões e multiplicações que são executadas quando se avalia `g(n)` é dado por $(n-1)+\sum_{k=2}^{n-1}(n-k)$ e, consequentemente, é $\frac{n^2-n}{2}$.

2. A função *Python* `h` tem comportamento similar ao da função `g`. Conclua que, para cada $n \in \mathbb{N}_2$, o número de multiplicações que são executadas quando se avalia `h(n)` é dado por $\sum_{k=1}^{n-1}(\sum_{1=1}^{k-1}1)$ e é $\frac{n^2-3n+2}{2}$.

3. A função *Python* `GaussElimination` é uma versão simplificada do algoritmo de eliminação de Gauss. Recebe uma matriz `A` invertível $n \times n$ e uma matriz `b` $n \times 1$, e devolve uma matriz `Aux` triangular $n \times n$ e uma matriz `baux` $n \times 1$ tais que os sistemas de equações `AX=b` e `Aux X=baux` são equivalentes, onde `X` é a matriz $n \times 1$ das variáveis (matrizes coluna são representadas pelas listas dos seus elementos). Assume-se ainda que `A` é tal que os diferentes denominadores envolvidos no algoritmo de eliminação são sempre não

nulos, não havendo assim necessidade de troca de linhas. Quando se avalia `GaussElimination[A,b]` quantas somas e subtracções são executadas? E quantas multiplicações e divisões?

```
def maximo(v):
  if len(v)==1:
      return v[0]
  elif v[0]>=maximo(v[1:]):
      return v[0]
  else:
      return maximo(v[1:])
```

Fig 4

```
def merge(u,v):
  res=[]
  i=0
  j=0
  for k in range(len(u)+len(v)):
   if i<len(u) and (j==len(v)
                    or u[i]<v[j]):
       res.append(u[i])
       i=i+1
   else:
       res.append(v[j])
       j=j+1
  return res
```

```
def mergesort(w):
    if len(w)<2:
       return w
    else:
       m=ceil(len(w)/2)
       w1=mergesort(w[:m])
       w2=mergesort(w[m:])
       return merge(w1,w2)
```

Fig 5

4. A função *Python* `maximo` calcula, recursivamente, o máximo de uma lista `v` não vazia de números. Conclua que quando se avalia `maximo(v)`, onde `v` é uma lista de comprimento $n \geq 1$, o número u_n de comparações do tipo `v[0]>=maximo(v[1:])` que são executadas no pior caso (isto é, quando `v` está ordenada por ordem crescente e os seus elementos são todos distintos) é tal que $u_1 = 0$ e $u_n = 1 + 2u_{n-1}$ para $n \geq 2$. Usando funções geradoras conclua que $u_n = 2^{n-1} - 1$ para cada $n \geq 1$.

5. A função *Python* `mergesort` (que usa a função auxiliar `merge`, à esquerda) corresponde a um dos mais conhecidos algoritmos para ordenação de listas de números (`ceil(x)`$=\lceil x \rceil$, isto é, o menor inteiro que é maior ou igual que `x`). É um algoritmo eficiente do tipo *dividir para conquistar*. Conclua que quando se avalia `mergesort(w)`, onde `w` é uma lista de comprimento n, um majorante c_n para o número de comparações que são executadas é tal que $c_1 = 1$ e $c_n = 2c_{\lceil \frac{n}{2} \rceil} + n$ para $n \geq 2$. Usando funções geradoras conclua que se n é uma potência de 2, então $c_n = n(1 + \log_2 n)$ para $n \in \mathbb{N}$. Sugestão:[1] comece por mostrar que, resolvendo a recorrência $s_0 = 1$ e $s_k = 2s_{k-1} + 2^k$ para cada $k \in \mathbb{N}_1$, se conclui que $s_k = (k+1)2^k$ para cada $k \in \mathbb{N}$, e obtenha depois o resultado pretendido considerando $c_{2^k} = s_k$.

2 Máximo divisor comum ($a \frown b$ ou mdc(a, b)) e algoritmo de Euclides

1. Use o algoritmo de Euclides para calcular o máximo divisor comum de

   (a) 74 e 44 (c) 92 e 36 (e) 84 e 136 (g) 1020 e 924

   (b) 84 e 54 (d) 75 e 108 (f) 644 e 826 (h) 2368 e 942

   Use a aplicação *WolframAlpha* recorrendo à função `gcd` (*greastest commom divisor*), para confirmar os resultados que obteve. Por exemplo, ao avaliar `gcd(74,44)` obtém a solução da alínea (a).

2. Calcule o máximo divisor comum de

   (a) 0 e -45 (d) 84 e -54 (g) -1020 e -924

   (b) 45 e -45 (e) -92 e 36 (h) -2478 e -240

   (c) -74 e 44 (f) 644 e -826 (i) -3649 e 129

3. Mostre que o número de divisões executadas na aplicação do algoritmo de Euclides a $m, n \in \mathbb{N}_1$ é menor ou igual que 5 vezes o número de dígitos do menor (e por isso se diz que a complexidade do algoritmo é linear no tamanho do seu *input*). Este resultado é habitualmente conhecido por *Teorema de Lamé*.

[1] Este exercício está resolvido, como exemplo, na secção 6.10 do livro.

3 Coeficientes de Bézout

Usando se necessário o algoritmo de Euclides estendido (algoritmo de Saunderson), calcule coeficientes de Bézout para a e b, e escreva o máximo divisor comum de a e b como combinação linear de a e b, quando

1. $a = 15$ e $b = 0$
2. $a = 0$ e $b = -12$
3. $a = 38$ e $b = -38$
4. $a = 74$ e $b = 44$
5. $a = 92$ e $b = 36$
6. $a = 534$ e $b = 84$
7. $a = 644$ e $b = 826$
8. $a = 1020$ e $b = 924$
9. $a = 92$ e $b = -36$
10. $a = -92$ e $b = 36$
11. $a = -534$ e $b = 84$
12. $a = -1020$ e $b = -924$

Use a aplicação *WolframAlpha*, recorrendo à função `ExtendedGCD`, para confirmar os resultados. Por exemplo, ao avaliar `ExtendedGCD[24,18]` obtém o resultado `{6,{1,-1}}`, que significa que o máximo divisor comum de 24 e 18 é 6 e que 1 e -1 são coeficientes de Bézout para 24 e 18, mais precisamente, que $1 \times 24 + (-1) \times 18 = 6$.

4 Miscelânea

1. Mostre que a soma de quaisquer dois números inteiros ímpares consecutivos é um múltiplo de 4.
2. Mostre que 8 é divisor de $a^2 - 1$, qualquer que seja número inteiro ímpar a.
3. Encontre 2 números naturais que somem 175 e cujo máximo divisor comum seja 25. (Livro: exemplo 8)
4. Encontre 2 números naturais cujo produto seja 6912 e cujo máximo divisor comum seja 12.
5. Sejam $m, n \in \mathbb{N}$ primos ente si. Sabendo que o seu produto aumenta 18480 quando triplicamos cada um deles, que m é múltiplo de 11 e que n é múltiplo de 10, determine m e n. (Livro: exercício na pág. 47)
6. Se dividirmos $m \in \mathbb{N}_1$ e $n \in \mathbb{N}_1$ pelo seu máximo divisor comum d, a soma dos quocientes obtidos é 10. Calcule m e n sabendo que $(m \times n)/d = 504$.
7. Mostre que dois números naturais consecutivos são sempre primos entre si.
8. Sejam $m, n \in \mathbb{Z}$ não simultaneamente nulos e $t \in \mathbb{Z}$. Mostre que se m e n são múltiplos de t então $\text{mdc}(m, n)$ também é múltiplo de t. Sugestão: use coeficientes de Bézout para m e n.
9. Sejam $m, n \in \mathbb{Z}$ não simultaneamente nulos e $t \in \mathbb{Z}$. Mostre que se $m \times n$ é múltiplo de t e $\text{mdc}(m, t) = d$ então n é múltiplo de t/d. Sugestão: use coeficientes de Bézout para m e t.
10. Sejam $m, n_1, n_2 \in \mathbb{Z}$. Mostre que se m e n_1 são primos entre si, e m e n_2 são primos entre si, então m e $n_1 \times n_2$ são também primos entre si. Sugestão: recorra a coeficientes de Bézout.
11. Mostre que se $\text{mdc}(m, n) = d$ $(m, n \in \mathbb{N})$ e $r \in \mathbb{N}_1$, então $\text{mdc}(rm, rn) = rd$. Sugestão: conclua primeiro que $rd \leq \text{mdc}(rm, rn)$, e, depois, recorrendo a coeficientes de Bézout para m e n, conclua que $rd \geq \text{mdc}(rm, rn)$.

5 Equações diofantinas (início)

1. Para cada uma das equações diofantinas seguintes, indique se tem ou não solução e, em caso afirmativo, calcule <u>uma</u> solução.

   (a) $4x + 8y = 22$
   (b) $3x + 5y = 14$
   (c) $4x + 6y = 8$
   (d) $20x + 12y = 2$
   (e) $44x + 56y = 24$
   (f) $92x + 36y = 20$
   (g) $-27x + 12y = 21$
   (h) $75x - 108y = 30$
   (i) $-5x + 8y = -9$

2. Indique o conjunto de todas as soluções das equações diofantinas do exercício anterior com solução.

   Use a aplicação *WolframAlpha* para confirmar os resultados. Por exemplo, se avaliar `solve 13x+8y=5 over the integers` obtém `y=13n+12 and x=-8n-7 n`$\in \mathbb{Z}$ que é a solução geral da equação diofantina $13x + 8y = 5$. Note que há várias formas equivalentes de apresentar a solução, e pode acontecer que as expressões que obteve não sejam exatamente iguais às apresentadas pelo *WolframAlpha*.