025089 – Projeto e Análise de Algoritmos

Aula 09

lizier@dc.ufscar.br

Questão 1

- As soluções da primeira coluna (algoritmos ***method1-2sum e method1-3sum***) testam todos os pares ou triplas possíveis do vetor, contando quais possuem soma zero.
- As soluções da segunda coluna (algoritmos ***method2-2sum e method2-3sum***) ordenam o vetor e buscam pelo último elemento (segundo no problema ***2sum*** e terceiro no problema ***3sum***), sendo esse último elemento o complemento da soma dos anteriores, para que a soma total seja zero.
- Complexidade:
  - ***method1-2sum:*** $\Theta(N^2)$
  - ***method1-3sum:*** $\Theta(N^3)$
  - ***method2-2sum:*** $\Theta(NlogN)$, se escolhermos uma ordenação $O(NlogN)$ e uma busca binária $O(logN)$
  - ***method2-3sum:*** $\Theta(N^2logN)$, se escolhermos uma busca binária $O(logN)$

Questão 2

- Complexidade:
  - ***POL-HORNER:*** $\Theta(k)$
    - $k+1$ multiplicações ou somas
  - ***POL:*** $\Theta(k^2)$
    - O problema aqui é que em cada iteração do for temos um método que não tem custo constante, como acontece no POL-HORNER. O método POT tem custo linear em relação a i, tornando o número de multiplicações em cada iteração: $1 + 2 + 3 + 4 + 5 \ldots \sim= n^2$
- O ***POL-HORNER*** é melhor que o ***POL***.

Questão 3

```
// int L[N][M], int N, int M
// int resp[N*M][2]
// primeira chamada: backtrack(0,0,0)

bool backtrack( int idx, int n, int m ) {
   L[n][m] = 2; resp[idx][0] = n+1; resp[idx][1] = m+1;
   if( n == N-1 && m == M-1) {
      for(int i=0; i<=idx; i++)
         cout << "(" << resp[i][0] << "," << resp[i][1] << ") ";
      return true;
   } else {
      if( n+1 < N && L[n+1][m]==0 && backtrack(idx+1,n+1,m) )
         return true; // baixo
      if( n > 0 && L[n-1][m]==0 && backtrack(idx+1,n-1,m) )
         return true; // cima
      if( m+1 < M && L[n][m+1]==0 && backtrack(idx+1,n,m+1) )
         return true; // direita
      if( m > 0 && L[n][m-1]==0 && backtrack(idx+1,n,m-1) )
         return true; // esquerda
   }
   L[n][m] = 0;
   return false;
}
```

Questão 3

- Complexidade:
  - O(3^(N*M))
    - 3 possibilidades em cada posição do labirinto (número máximo de filhos de cada nó da árvore de chamadas)
    - Máximo de N*M posições no labirinto

Questão 4

- Complexidade:
  - ***a)*** *$\Theta(N^2)$, 0 + 1 + 2 + 3 + ... + N comparações*
  - ***b)*** *$\Theta(logN)$*
  - ***c)*** *$\Theta(KlogK)$*