MAC121 - Algoritmos e Estruturas de Dados I

Universidade de São Paulo

Segundo Semestre de 2020

Recursão e backtracking

Busca recursiva

Backtracking recursivo

Podemos usar recursão para implementar backtracking (sem o uso da pilha).

Problema: Escreva uma função recursiva com protótipo

```
int nRainhasRec (int **tab, int n, int atual);
```

que devolve 1 se é possível colocar as n rainhas num tabuleiro $n \times n$ a partir da rainha na linha *atual* (com as anteriores já colocadas) e 0 caso contrário.

Busca recursiva

Problema: Faça uma função recursiva

```
int busca (int *v, int n, int x);
```

que devolve um índice i tal que $v[i] = x$ ou -1, se x não ocorre no vetor.

Problema: Faça uma função recursiva

```
int buscabin (int *v, int ini, int fim, int
x);
```

que recebe um vetor ordenado e índices *ini* e *fim* e devolve um índice i no intervalo $[ini, fim]$ tal que $v[i] = x$ ou -1, se x não ocorre no intervalo do vetor.

A notação $O()$

Em Computação usamos a notação assintótica $O()$ para denotar a complexidade (de tempo ou espaço usado) de algoritmos. De forma intuitiva, a notação dá uma ideia da complexidade, sem nos preocuparmos com constantes ou fatores menores.

Exemplos:

- Se um algoritmo executa, para toda entrada com n elementos, $4n^2 - 10n + 123$ passos, dizemos que este algoritmo é $O(n^2)$.
- Se executa $n + 100 \log n + 1239$ passos, é $O(n)$.
- Se a entrada tem parâmetros m e n e executa $5mn^2 - 3mn + m \log n$, será $O(mn^2)$.

Notação $O()$

Vamos usar isso informalmente nesta disciplina. A ideia é que a função completa, quando olhamos para valores grandes dos parâmetros (assintoticamente), tem comportamento parecido com a função mais simples.

Dessa forma, dizemos que a busca linear num vetor com n elementos tem consumo de tempo $O(n)$, enquanto que a busca binária, tem consumo $O(\log n)$.

Para saber mais

- Material do P. Feofiloff sobre backtracking, recursão e algoritmos de enumeração
- Material do P. Feofiloff sobre busca