Análise (informal) de Algoritmos

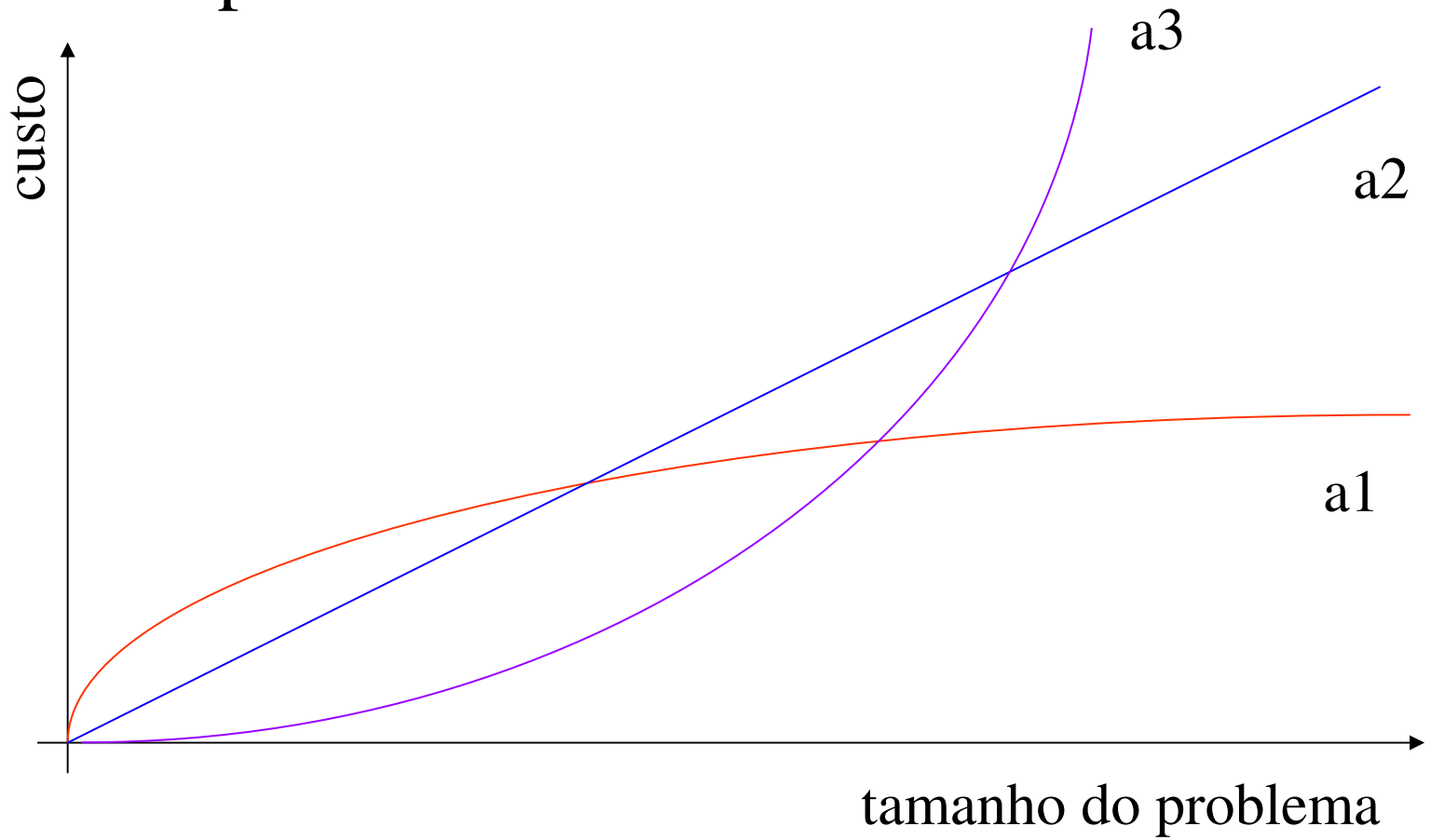
- Dado um algoritmo para resolver um determinado problema, é importante que se tenha mecanismos objetivos de avaliação do mesmo.
- A avaliação de um algoritmo deve ser independente de
  - implementação numa linguagem
  - máquina onde o mesmo é executado

Análise (informal) de Algoritmos

- Critérios
  - custo (tempo e memória)
  - 'tamanho' do problema
  - comportamento
  - operações críticas

Análise (informal) de Algoritmos

- Comportamento



Um exemplo: bubblesort

```
void bubblesort(int v[], int n){
  int i, k, t;
  do{
    k = 0;
    for(i=0; i < (n-1); i++)
      if(v[i] > v[i+1]) {
        t=v[i]; v[i]=v[i+1]; v[i+1]= t;
        k++;
      }
  } while (k > 0);
}
```

Um exemplo: bubblesort

- Operação: comparação *
- A cada execução do comando "for" pelo menos um elemento é colocado na sua posição definitiva, portanto esse comando será executado no máximo n vezes.
- Cada execução do comando "for" realiza n-1 comparações.
- O número total de comparações é
  - pior caso: n*(n-1) ou n^2-n.
  - melhor caso: n-1

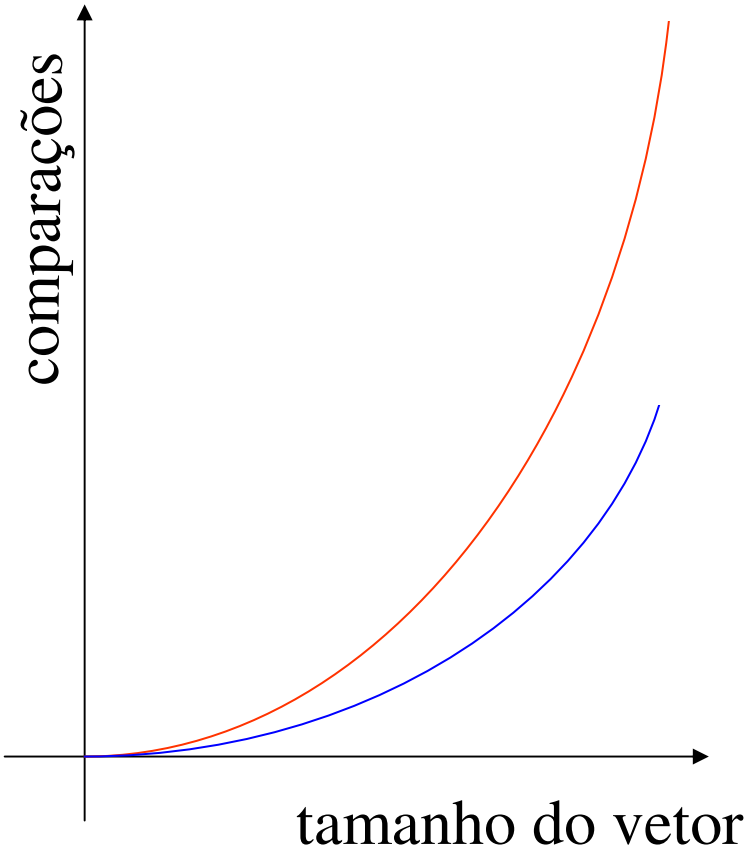
Bubblesort melhorado

```
void bubblesort(int v[], int n){
  int i, k, t, m = n-1;
  do{
    k = 0;
    for(i=0; i < m; i++)
      if(v[i] > v[i+1]) {
        t=v[i]; v[i]=v[i+1]; v[i+1]= t;
        k++;
      }
    m--;
  } while (k > 0);
}
```

Bubblesort melhorado

- A cada execução do comando "for" o trecho do vetor a ser percorrido diminui de um.
- Número de comparações:
  - pior caso: (n-1)+
    (n-2)+...+1 = ((n*(n-1))/2=(n^2-n)/2
  - melhor caso: n-1

Comparando as duas versões



Número de comparações no pior caso:

- versão 1: n^2-n
- versão 2: (n^2-n)/2

Nos dois casos, o termo predominante, é n^2

Limite inferior

- Para alguns problemas, é possível determinar o comportamento do 'melhor algoritmo possível', sem mesmo ter esse algoritmo.
- Exemplo: dado um número inteiro k e um conjunto de inteiros C, determinar se C contém k.
  - Qualquer que seja o algoritmo, ele terá que comparar k com todos os elementos de C.

Um exemplo

Ordenação de uma seqüência de n elementos, baseada em comparações:

- Para n valores distintos podemos ter **n!** sequências diferentes.
- Uma comparação na melhor das hipóteses divide o número de possibilidades pela metade.
- O melhor algoritmo de ordenação portanto irá fazer $\mathbf{\log_2(n!)}$ comparações $\mathbf{\sim n \log_2(n)}$

Classificação dos algoritmos

- A maior parte dos algoritmos que veremos no curso pode ser 'enquadrada' nas seguintes categorias (n é o tamanho do problema), segundo a expressão que descreve o seu comportamento:
- logarítmicos: proporcional a log n
- lineares: proporcional a n
- polinomiais: proporcional a n^k
  - quadrático: proporcional a n^2
- exponenciais: proporcional a k^n

Classificação dos problemas

- Decidíveis: problemas que podem ser resolvidos por um algoritmo (um algoritmo é por definição finito).
- Indecidíveis: problemas para os quais não é possível construir um algoritmo para resolver.