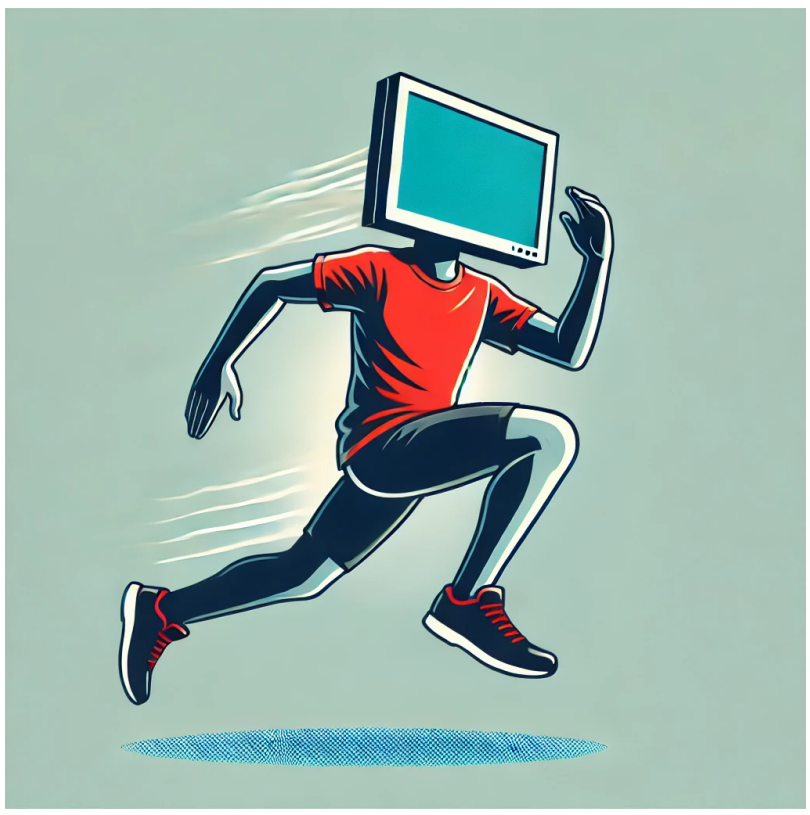
Update UT3Problema2.md
José R. Mas Davó authored 4 weeks ago

8563d333

UT3Problema2.md 4.08 KiB

UT3 - Problema 2: Anàlisi d'eficiència d'algoritmes

Agrupament: Individual



Pregunta guia

Com podem mesurar i comparar l'eficiència dels algoritmes de cerca i ordenació en arrays de diferents grandàries per a determinar quin és més adequat en diverses situacions?

Context i descripció del problema

Un equip de desenvolupament de software està treballant en l'optimització d'una aplicació que maneja grans quantitats de dades. Necessiten determinar quins algoritmes de cerca i ordenació són més eficients per a diferents volums de dades. La vostra tasca és desenvolupar un programa que mesure el temps d'execució de diversos algoritmes de cerca i ordenació, utilitzant arrays de diferents grandàries, i analitzar els resultats per proporcionar recomanacions basades en l'eficiència.

El programa ha de comparar:

1. Els algoritmes de cerca lineal i binària en arrays ordenats de 1.000, 10.000, 100.000 i 1.000.000 elements.
2. Els algoritmes d'ordenació d'inserció directa, selecció directa i intercanvi directe (bombolla) en arrays desordenats de 100, 1.000, 10.000 i 100.000 elements.

Objectius d'aprenentatge

En completar aquesta pràctica, sereu capaços de:

- Implementar i aplicar algoritmes de cerca (lineal i binària) en arrays.
- Implementar i aplicar algoritmes d'ordenació (inserció directa, selecció directa i intercanvi directe) en arrays.
- Utilitzar mètodes per mesurar el temps d'execució en Java.
- Analitzar i comparar l'eficiència d'algoritmes basant-se en el temps d'execució.
- Avaluar la relació entre la grandària de les dades i el rendiment dels algoritmes.
- Generar i manipular arrays de grans dimensions amb dades aleatòries.

Instruccions

1. **Investigació preliminar:**

- Investiga sobre els algoritmes de cerca lineal i binària.
- Investiga la implementació dels algoritmes d'ordenació d'inserció directa, selecció directa i intercanvi directe (bombolla).
- Investiga com mesurar el temps d'execució en Java utilitzant `System.currentTimeMillis()`.

2. **Desenvolupament del programa:**
   - Crea arrays amb les grandàries especificades i inicialitza'ls adequadament per a cada prova.
   - Implementa els algoritmes de cerca lineal i binària, i els tres algoritmes d'ordenació.
   - Mesura el temps d'execució per a cada combinació d'algoritme i grandària d'array.
3. **Execució i anàlisi:**
   - Executa el programa per obtenir els temps d'execució per a cada algoritme i grandària d'array.
   - Analitza els resultats per determinar:
     - Quin algoritme de cerca és més eficient per a cada grandària d'array.
     - A partir de quina grandària d'array la cerca binària esdevé més eficient que la lineal.
     - Quin algoritme d'ordenació és més eficient per a cada grandària d'array.
   - Crea gràfics o taules per a visualitzar els resultats.
4. **Reflexió i documentació:**
   - Escriu un informe breu que incloga:
     - Els resultats obtinguts, presentats en forma de taules o gràfics.
     - Una anàlisi dels resultats, explicant per què certs algoritmes són més eficients en determinades situacions.
     - Recomanacions sobre quin algoritme utilitzar segons la grandària de les dades.

Notes d'ajuda

Per a mesurar el temps d'execució d'un bloc de codi en Java:

```java
long tempsInicial = System.currentTimeMillis();
/* Ací el codi de la tasca */
long tempsFinal = System.currentTimeMillis();
long tempsExecucio = tempsFinal - tempsInicial;
System.out.println("Temps d'execució: " + tempsExecucio + " ms");
```

Entrega

- Envieu el fitxer del codi font (`UT3Problema2.java`) i l'informe en format PDF.
- Recorda seguir la convenció de noms.
- **Important:** Afig el teu nom i cognoms sempre com a comentari al principi dels fitxers JAVA (En NetBeans després de @author).

Extensió opcional

Per als estudiants que acabin abans o vulguin un repte addicional:

- Implementa i compara algoritmes d'ordenació més avançats com *Quicksort* o *Mergesort*.