Resum

Objectiu:
L'objectiu d'aquest estudi és analitzar la diferència d'eficiència entre els algorismes Selection Sort (S) i Insertion Sort (I) en l'ordenació de vectors de mides desiguals amb valors aleatoris. Es vol determinar quin dels dos algorismes, S o I, resulta més eficient en termes de temps d'execució quan s'aplica a conjunts de dades de diferents dimensions.

Mètode: Hem creat un programa en C++ per generar 50 vectors aleatoris amb valors aleatoris entre 0 i 10^6, i amb una mida aleatòria que oscil·la entre 1000 i 5000 elements. En el mateix programa, aplicarem els algorismes Selection Sort i Insertion Sort en ordre aleatori per a cadascun d'aquests vectors, i mesurarem el temps d'execució de cada procés per avaluar-ne l'eficiència.

Resultats:
Hem obtingut la mitjana de temps del Selection Sort és de 0.01505916s, mentre que la mitjana de temps del Insertion Sort és de 0.01105572s. Això implica una diferència mitjana de temps entre els dos algoritmes de 0.00400344s.
Per avaluar aquesta diferència de manera estadística, hem treballat amb la transformació logarítmica dels temps d'execució. La mitjana de la diferència dels logaritmes naturals dels temps és 0.308682, amb un interval de confiança del 95% que va de 0.3026156 a 0.3147483. I l'estadístic de prova obtingut per aquesta anàlisi és 102.26, un valor, juntament amb un p-value <2.2e-16.

Conclusió:
Amb un IC del 95%, els resultats de l'anàlisi estadística ens permeten afirmar amb seguretat que el temps d'ordenació de l'algoritme Insertion Sort és diferent del de Selection Sort.

La relació entre els temps d'ordenació, representada com S/I, ens proporciona una mesura clara de la magnitud d'aquesta diferència. En termes mitjans, el Selection Sort triga aproximadament 1.3616 vegades més que el Insertion Sort a ordenar els vectors del conjunt de dades analitzat. Amb un interval de confiança del 95%, aquesta relació es troba en el rang de [1.3534,1.3699], fet que demostra que la diferència és consistent i significativa.

Per tant, podem concloure que, en la majoria de casos, l'algoritme Insertion Sort serà més ràpid que el Selection Sort a l'hora d'ordenar vectors.

Introducció:
A l'assignatura EDA que estem cursant aquest quadrimestre, utilitzem diferents tipus d'algorismes, i ens ha sorgit la idea d'analitzar quin dels dos algorismes d'ordenació, Selection Sort (S) o Insertion Sort(I) (ambdós de complexitat $O(n^2)$), és més eficaç.

Volem conèixer si existeix una diferència d'eficiència entre els dos algorismes en casos aleatòries per ordenar vectors de diferents mides. Es vol determinar quin dels dos algorismes resulta més eficient en termes de temps d'execució quan s'aplica a conjunts de dades de diferents dimensions. Aquesta comparativa ens permetrà establir quin dels mètodes és més adequat per ordenar dades aleatòries segons la mida del vector.

Mètodes:

Recollida de dades:

Per a la recollida de dades aleatòries, hem creat un programa en C++ per generar 50 vectors aleatoris amb valors entre 0 i 10^6, i amb una mida també aleatòria entre 1000 i 5000 elements. En el mateix programa, aplicarem els algorismes de Selection Sort i Insertion Sort a cadascun d'aquests vectors, i mesurarem el temps d'execució.

Aquest treball es realitzarà en un Virtual Machine amb un processador AMD Ryzen 3600X 3.8GHZ, una memòria Ram de 16GB i un disc dur HDD de 1TB, sota un sistema operatiu Linux. Com que ambdós algorismes tenen una complexitat temporal de O(n^2), recollirem els temps d'execució per estudiar la seva eficiència, analitzant si hi ha diferències significatives entre ells i com varia el temps d'execució en funció de la mida del vector.

Variables recollides:

La nostra variable d'interès Y, és el temps d'ordenació d'un algorisme, mesurat en segons. La variable X representa el tipus d'algorisme utilitzat per ordenar els vectors, que en aquest cas pot ser Selection sort o Insertion sort. Finalment, la variable Z correspon a la mida del vector a ordenar.

Anàlisi estadística

1. Permisses:

-Mostres aleatòries aparellades:

Per obtenir una mostra aleatòria simple de la població de vectors a ordenar, hem decidit crear vectors de diferents mides generats aleatòriament. Així, tant la mida com els elements de cada vectors seran generats aleatòriament, formant una mostra de 50 vectors amb mides diferents. El disseny de l'estudi és amb dades aparellades, ja que cadascun dels vectors serà ordenat mitjançant ambdós algorismes, Selection Sort y Insertion Sort, per tal de comparar els temps d'execució dels dos algorismes en cada cas.

- La premissa de normalitat

2. Càlcul de l'estadístic:

$$t=\frac{(\bar{d}-\mu_0)}{s_d/\sqrt{n}} \quad t_{n-1} \qquad\qquad IC(d,1-\alpha)=[\bar{d} \mp t_{n-1,0.975}\cdot se]$$

La mitjana $\bar{d}$ representa el valor mitjà de les diferències entre els temps d'execució dels dos algoritmes, és a dir, el rendiment relatiu mitjà d'un respecte a l'altre. Per la seva banda, s_d és la desviació estàndard de $\bar{d}$, se és l'error estàndard. Finalment, n fa referència a la mida de la mostra, en aquest cas 50 vectors, que determina la quantitat de parelles de temps analitzades per comparar els dos algoritmes.

L'estadístic t que utilitzem per a la nostra prova estadística segueix una distribució t-Student amb 49 graus de llibertat, ja que treballem amb una mostra de 50 observacions. A més, hem fixat un nivell de confiança del 95%, cosa que implica un risc d'error del 5% (α=0.05). Si p-value Si el p-value és menor que α, això indicaria que existeix evidència suficient per concloure que hi ha una diferència significativa entre els temps d'execució dels dos algorismes.

Alternativament, podem justificar que hi ha una diferència en l'eficiència d'ordenació si el valor absolut de l'estadístic t, $|t_{n-1}|$, és més gran que el valor crític $t_{49,0.975}$., corresponent al percentil 97.5 de la distribució t-Student amb 49 graus de llibertat.

Resultats:

Descriptiva:
La Taula 1 mostra les mitjanes i les desviacions estàndards dels temps d'ordenació per a cada algorisme, així com les diferències entre els temps d'execució, expressades en segons.

Tipus de algorisme	Nº d'execucions	Mitjana de temps	Desviació estàndard
Insertion Sort(I)	50	0.01105572	0.007035966
Selection Sort(S)	50	0.01505916	0.009618559
Diferencia (S-I)	50	0.00400344	0.002593511

Taula 1. Estadístics mostrals de les dades

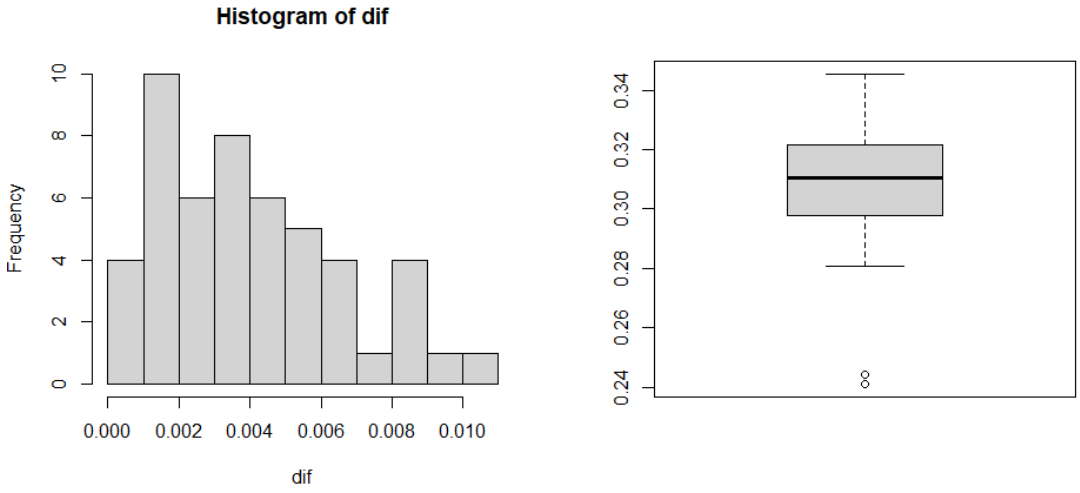

Figura.1 gràfic de hist(dif) i boxplot(dif)

Observant la Taula 1, podem veure que es presenten la mitjana de temps d'ordenació i la desviació estàndard per a ambdós algorismes, així com també la mitjana i la desviació estàndard de les diferències entre els temps d'execució de Selection Sort i Insertion Sort. A partir d'aquestes dades, juntament amb els histogrames mostrats a la Figura 1, que representen la distribució de les diferències entre els temps d'ordenació dels dos algorismes, podem observar que hi ha una diferència significativa entre la mitjana dels temps d'execució de Selection Sort i Insertion Sort.
Els resultats indiquen que, en general, la mitjana de temps d'ordenació del Selection Sort és notablement més gran que la de l'Insertion Sort, la qual cosa suggereix que l'Insertion Sort és més eficient en termes de temps d'execució per ordenar vectors de mides aleatòries.

La premissa de normalitat

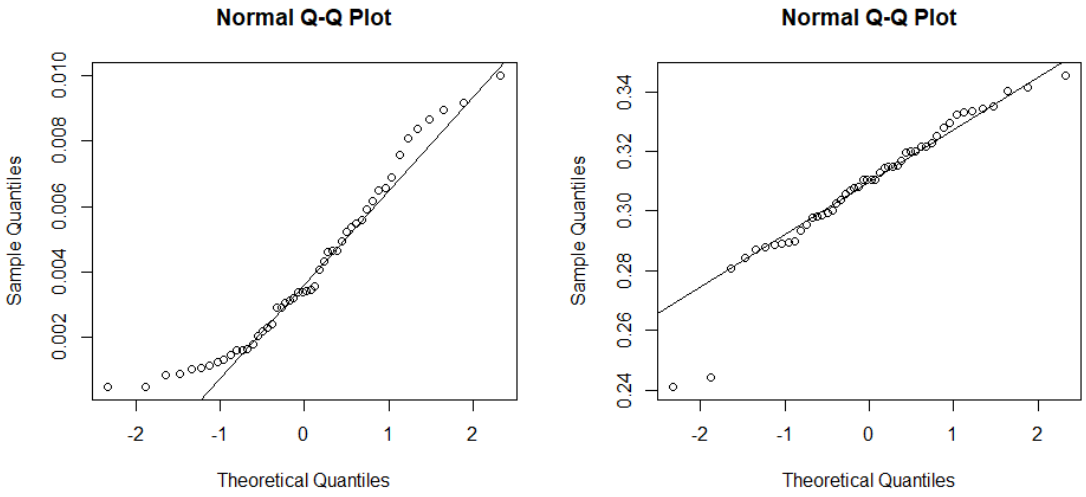

Figura 2

Comprovació de La premissa de normalitat amb qqplot.

En el gràfic situat a l'esquerra de la Figura 2, es mostren els quantils empírics de la diferència de temps d'ordenació S-I enfront dels quantils teòrics d'una distribució normal. Podem observar que els quantils empírics no segueixen la recta de regressió dels quantils teòrics de manera precisa. Davant d'aquesta manca de normalitat, hem decidit aplicar una transformació logarítmica als temps d'ordenació. El gràfic de la dreta de la Figura 2 mostra els quantils empírics de les diferències transformades enfront dels quantils teòrics d'una distribució normal. En aquest cas, els quantils empírics s'ajusten molt millor a la recta de regressió, la qual cosa indica que, després de la transformació, les diferències de temps d'ordenació segueixen més de prop una distribució normal.

```
> t.test(logsele,loginser,paired=TRUE)

        Paired t-test

data:  logsele and loginser
t = 102.26, df = 49, p-value < 2.2e-16
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 0.3026156 0.3147483
sample estimates:
mean difference
       0.308682
```

Observem que el p-value obtingut és < 2.2e-16, un valor molt més petit que el nivell de significació establert (α=0.05). Aquest resultat posa en evidència que l'algorisme Selection Sort tendeix a ser menys eficient que Insertion Sort en termes de temps d'execució, segons les dades analitzades. Aquesta conclusió és coherent amb l'anàlisi gràfica prèvia, que ja suggeria una diferència significativa en les distribucions de temps dels dos algoritmes.

Tenim que la mitjana de la diferència del logaritme natural dels temps d'ordenació, ln(S/I), és 0.308682. A més, disposem d'un interval de confiança del 95% per aquesta mitjana, que es troba entre 0.3026156 i 0.3147483. A partir d'aquestes dades, podem calcular quantes vegades tarda més el Selection Sort en comparació amb l'Insertion Sort, interpretant els valors obtinguts en termes de la relació S/I.
Per determinar aquesta relació, utilitzem la fórmula:
$\ln(S/I) = 0.308682 \Rightarrow S/I = e^{0.308682} = 1.3616$
Això significa que, de mitjana, el Selection Sort triga aproximadament 1.3616 vegades més que l'Insertion Sort.

A continuació, apliquem aquesta mateixa transformació a l'interval de confiança per obtenir l'interval de confiança per a S/I
Apliquem:
$IC(S/I,0.95) \Rightarrow [e^{0.3026156}, e^{0.3147483}] = [1.3534,1.3699]$.

Aquest interval indica que, amb un 95% de confiança, Selection Sort tarda entre 1.3534 i 1.3699 vegades més que Insertion Sort en vectors amb valors aleatoris entre 0 i 10^6, i amb una mida que oscil·la entre 1000 i 5000 elements.

Discussió

Interpretació dels resultats de les prediccions

Nosaltres hem analitzat la diferència d'eficiència entre els algorismes Selection Sort (S) i Insertion Sort (I) aplicats a conjunts de vectors aleatoris. Si considerem un vector de mida 3000 elements, per exemple, i volem determinar quin algorisme serà més eficient en termes de temps d'execució, podem utilitzar les prediccions basades en els resultats obtinguts.

La mitjana de temps d'execució del Selection Sort és de 0.01505916 segons, mentre que la del Insertion Sort és de 0.01105572 segons. Això implica que, de mitjana, el Selection Sort trigarà aproximadament 1.3616 vegades més que el Insertion Sort per ordenar aquest vector, amb un 95% de confiança que aquesta relació es trobi entre [1.3534, 1.3699].

Aquestes dades suporten la hipòtesi inicial que l'Insertion Sort és més eficient en termes de temps d'execució, especialment en conjunts de dades similars als generats en aquest estudi. La relació S/I també confirma que, fins i tot amb un interval de confiança, la diferència d'eficiència es manté consistentment a favor de l'Insertion Sort.

Limitacions de l'estudi

Una de les principals limitacions d'aquest estudi és la manca de diversitat en els conjunts de dades generats. Tot i que els vectors són aleatoris en mida i valors, estan limitats a un rang fix de 1000 a 5000 elements i a valors entre 0 i 10^6. Això pot restringir la generalització dels resultats a altres casos o dimensions de vectors.

A més, el fet que els resultats es basen en una màquina específica (un Virtual Machine amb un processador AMD Ryzen 3600X i 16GB de RAM) pot implicar que els temps d'execució podrien variar significativament en altres plataformes amb diferents configuracions de maquinari.

Finalment, els algorismes comparats tenen complexitats similars O(n2)O(n2), i per tant, l'estudi no inclou alternatives amb eficiències superiors com QuickSort o MergeSort, que podrien ser més adequades en casos pràctics amb dimensions més grans o estructures de dades més complexes.

Conclusió

Partint dels resultats obtinguts, queda clar que l'Insertion Sort és, en la majoria de casos, més eficient que el Selection Sort en termes de temps d'execució. No obstant això, les limitacions exposades haurien de ser considerades per a futures investigacions, especialment en estudis que busquin explorar algorismes d'ordenació més avançats o aplicacions en entorns més diversificats.