1 Algoritm för att sortera data

Det är viktigt att du läser igenom hela instruktionen innan du börjar göra något.

1.1 Introduktion Sökmotor

De så kallade sökmotorerna på internet (google, bing, baidu, duckduckgo osv.) söker inte direkt på internet när du anger en sökterm. I stället går företagen ständigt igenom internet och kategoriserar (indexerar) sidor och skapar databaser; utifrån dessa sker sedan sökningen. Att sortera internet är en förutsättning för att du sedan ska kunna söka. Icke indexerade sidor finns på deep-web; en del av denna är dark-web.

1.2 Tidskomplexitet

Att sortera data är en stor sak i programmeringsvärlden. De enkla sorteringsalgoritmerna har tidskomplexiteten $O(n^2)$. De snabbare har tidskomplexiteten $O(n \log n)$, t.ex Merge sort.
Läs dokumentet Tidskomplexitet där dessa begrepp förklaras.

1.3 Urvalssortering

Urvalssortering (eng. Selection sort) finns beskrivet i dokumentet Urvalssortering_algoritm.

1.3.1 Uppgift

Om du vill ha stöd så gör uppgiften stegvis 1.3.1 och 1.3.2 och 1.4.1. Du kan också direkt ge dig på själva huvuduppgiften som är 1.4.1. Du bör fundera noga på att skriva def:ar och göra loopar inuti loopar.

Skriv kod som kan ta en lista och skriva ut den sorterad med hjälp av Urvalssortering. Använd listorna L1=[3,1,5,2,9,7,4], L2=[3,1,5,2,9,7,2] och L3=[0,0,0,0]. Testa gärna på fler.

1.3.2 Uppgift

Slumpmässiga listor kan genereras av: L=random.sample(range(a, b), n) som ger n heltal, alla olika,

från a (inkl) till b (exkl). Antalet n kan inte vara större än antalet tal mellan a och b. Exempelvis för random.sample(range(0, 10), n) kan n inte vara större än 10 (men 10 går bra för 0...9 är 10 tal).

Skriv kod som kan ta en slumpmässigt genererad lista och skriva ut den sorterad med hjälp av Urvalssortering.

Om man sätter b=n vet man exakt vilka tal som ingår så om b=10=n så finns i listan talen 0...9. Då skulle man i stället kunna söka efter 9, sedan efter 8 osv. Men det är inte Selection sort. Sätt hellre b=1e6 och n ska stegas från 1000 till 10 000.

1.4 Tidsmätning

Att mäta tiden för sortering.

För att mäta tid behöver vi importera modulen time.
Precis innan själva sorteringen börjar så skriv koden: start=time.time(). Observera att själva konstruktionen av listan ska inte vara med i tidmätningen.
När sorteringen är klar så skriv: end=time.time(). Beräkna tiden som end-start.
Har inte kontrollerat det men har för mig att det finns en nedre tidsgräns. Metoden är inte helt riskfri och kan störas om andra processer belastar CPU:n.

Behandla data från tidmätningen som om det vore en *statistisk undersökning*. En sortering av en lista med tex. 1000 tal är bara *en* mätning, för ett bra värde för sorteringstid för n=1000 måste flera listor med samma längd skapas slumpmässigt och sedan medeltid för sortering av lista med längd n=1000 beräknas. Så bilda ett medelvärde för n=1000, ett för n=1500 osv upp till 10000 tal i en lista. Sätt medeltiden på y-axeln och listlängden på x-axeln.
Rita graf och rita även in en andragradskurva $\left(y = Ax^2 \right)$ så det framgår att dina data följer en andragradskurva.

Mätningarna kopplar till texten om tidskomplexitet. Med Urvalssortering bör tidskomplexiteten öka kvadratiskt med längden, n, på listan som ska sorteras.

1.4.1 Uppgift

Redogör för resultat och problem och jämför med teorin angående Tidsmätning. Rita graf över hur sorteringstiden ökar med antalet tal som ska sorteras; anpassa den teoretiska kurvan och rita den och jämför.

1.5 Lite råd och tips.

Du ska *inte* använda kommandot max() som letar upp det största talet i en lista utan du ska skriva ett program som går igenom listan och letar upp det största talet osv.
Du ska *inte* använda kommandot sort() som sorterar en lista; då löser python uppgiften, inte du.

Under konstruktionen av programmet använder du naturligtvis lite mindre listlängder, sorteringen för 10000 tar tid.

Index är alltid ett problem, håll reda på dem.
Hur vet man när man är klar med sorteringen?
Om man skriver väldigt dålig kod så kan det ge oönskade effekter på tidskomplexiteten.

Ibland kan det vara bra att dela upp problemet i delproblem. Ett delproblem i Urvalssortering är att gå igenom listan och leta upp det största talet. Lös först det problemet; ange listor av olika sort och kontrollera att programmet verkligen hittar det största talet.

Bestämningen av konstanten A ska göras från datapunkterna från körningarna, den kan inte vara ett fast värde. Du kan tex. använda de 2 sista punkterna, kalla dem (x_1, y_1) och (x_2, y_2). Beräkna $A_1 = y_1 / x_1^2$ och $A_2 = y_2 / x_2^2$ och sedan $A = (A_1 + A_2)/2$. Det finns anpassningskommandon i python men de behöver du inte använda.

Funktionen $y = Ax^2$ är en andragradsfunktion. Mer allmänt kallas funktioner med x^n för potensfunktioner (exponenten är en konstant, basen varierar). Blanda inte ihop dem med a^x som är exponentialfunktioner (basen är en konstant, exponenten varierar).