Inledning
Algoritmen jag har valt att studera är Horspools algoritm för strängmatchning. Horspools algoritm är en effektiv strängmatchningsalgoritm som är baserad på Boyer-Moore-algoritmen. Horspools algoritm är en av de mest använda strängmatchningsalgoritmerna och är särskilt lämplig för att matcha långa strängar.
Huvudsyftet med laborationen är att studera algoritmens komplexitet men också att studera och implementera Horspools algoritm och att jämföra den med naiv brute-force lösning.

Algoritmen
Algoritmen använder sig av en "hopp-tabell" som bestämmer hur mycket den ska hoppa framåt i text-strängen vid varje steg. hopp-tabellen innehåller information om hur långt det är mellan olika tecken i text-strängen. Detta används för att bestämma hur mycket algoritmen ska hoppa framåt i text-strängen vid varje steg.

Enkelt förklarat generas hopp-tabellen såhär

- En hopp-tabell genereras för alla ascii värden(255).
- Tabellen fylls med längden av mönstret som standardvärde
- Om ett tecken i tabellen finns i mönstret ta längden minus positionen i mönstret minus ett
- Om tecknet är det sista i mönstret så sätts det lika med längden av mönstret

Själva algoritmen börjar på det sista tecknet i mönstret och jämför det med tecknet på samma position i text-strängen. Om tecknen matchar varandra, går algoritmen ett steg tillbaka i båda strängarna och jämför nästa par tecken. Detta fortsätter tills antingen ett tecken inte matchar eller tills algoritmen har nått början av mönstret. Om algoritmen når början av mönstret utan att ha hittat några olika tecken betyder det att mönstret finns i text-strängen. Algoritmen returnerar då positionen för första tecknet i mönstret i text-strängen. Om algoritmen inte når början av mönstret innan den hittar ett olika tecken, används informationen i hopp-tabellen för att bestämma hur mycket algoritmen ska hoppa framåt i text-strängen. Algoritmen fortsätter sedan att söka igenom text-strängen från den nya positionen.

Designval
I detta program har bara arrays av olika datatyper använts det har inte funnits någon fördel med att lagra texten och mönstret som träd eller någon annan datastruktur.

Det finns några olika val man kan göra avseende datastrukturen för hopptabellen. Det är inte nödvändigt att som jag gjort, ha ett element för varje tecken i ASCII-tabellen (0-255). Man kan istället bara ha en hopptabell för de tecken som faktiskt finns i mönstret, och alla andra tecken får samma hopplängd som för det sista tecknet i mönstret. Det skulle ge lite snabbare sökning, men det skulle också innebära att det blir mer besvärligt att generera hopptabellen.

Experimentplanering
Har designat experimentet genom att utföra en del körningar då mönstret är i början, slutet, mitten. Men för att få en bra jämförelse mellan BruteForce lösningen och Horspools algoritmen ska det genomföras flera körningar med slumpmässiga arrays med olika längder samt med och utan nyckeln i arrayen för att därefter få ett medelvärde. Samt testa visa specialfall då texten har flera matchande delar till mönstret för att fånga in eventuella Worst och Best cases.

Resultat
1. Specialfall

(Matchande delar igenom hela texten, Mönstret i slutet)
--Körning Specialfall 1--
Choose a text: abcabcabcabcabcabcabcaabc
Choose a pattern: aabc
Pattern found at pos: 20
Horspool Basic operations performed: 30
BruteForce Basic operations performed: 25

(Mönstret i början)
--Körning Specialfall 2--
Choose a text: trädstruktur
Choose a pattern: träd
Pattern not found in text

Horspool Basic operations performed: 4
BruteForce Basic operations performed: 4

2. Fix längd

Körning med fix längd på text och mönster, 100 iterationer
--Körning 1--
Choose your option: 3
Choose length of text: 30
Choose length of pattern: 5
Choose number of iterations: 100
Horspool Average basic operations performed: 7
BruteForce Average basic operations performed: 19

--Körning 2--
Choose your option: 3
Choose length of text: 50
Choose length of pattern: 5
Choose number of iterations: 20
Horspool Average basic operations performed: 10
BruteForce Average basic operations performed: 31

--Körning 3--
Choose your option: 3
Choose length of text: 50
Choose length of pattern: 40
Choose number of iterations: 20
Horspool Average basic operations performed: 21
BruteForce Average basic operations performed: 50

3. Körning med varierande text och mönster längd

(Mönstret finns inte med i vissa fal)
--Körning--
Choose your option: 4
Choose number of iterations: 200
Horspool Average basic operations performed: 8

BruteForce Average basic operations performed: 24

Analys
Det finns specialfall då BruteForce lösningen är snabbare än Horspool algoritmen och det inträffar när mönstret har matchande delar igenom hela texten, Detta är Worst Case för horspool.

Det finns också fall när horspool och bruteforcelösningen presterar lika bra, Det är då horspool inte nyttjar sin förmåga att hoppa fram i texten. Det inträffar när mönstret finns i början. Detta är även Best Case för horspool algoritmen.

Utifrån testkörningar med slumpmässiga arrays med varierande längd finns det två starka faktorer som påverkar tidskomplexiteten för Horspool algoritmen ett klart samband mellan längden av texten och längden av mönstret som påverkar tidskomplexiteten för horspool algoritmen. För det första, ju längre texten är, desto längre tar det algoritmen att söka igenom texten. För det andra, ju längre mönstret är, desto längre tar det algoritmen att söka igenom mönstret. Det återkopplar starkt till tidskomplexiteten för algoritmen O(n * m), där n är längden på texten och m är längden på mönstret.

Slutsatser och diskussion
Utifrån resultaten går det enkelt att konstatera att Horspool-algoritmen är betydligt bättre än BruteForce lösningen. Två till tre gånger färre kritiska operationer än BruteForce-algoritmen, Detta tack vare att Horspool-algoritmen använder sig av sin hopp-tabellen för att kunna hoppa över bokstäver som inte finns i mönstret. Då slipper den att göra dem tidskrävande jämförelse operationer som den naiva lösningen på problemet gör. Det finns såklart specialfall som sticker ut som gör horspool algoritmen långsam men dessa fall inträffar extremt sällan så dem är helt klart försumbara.