Informatik 5. Klasse
Examensvorbereitung
Teil 3

Suchen: Aufgabe 1

Schreibe eine Python-Funktion `linearsearch(L, item)`, welche die Positionen (Indizes), an denen `item` in `L` vorkommt, als Liste zurückgibt.

Beispiele:

```
L = [1, 0, 1, 1, 0]
print(linearsearch(L, 1)) # => [0, 2, 3]
print(linearsearch(L, 2)) # => []
```

Suchen: Aufgabe 2

Zeige schrittweise, wie die binäre Suche das Element 28 im Array A findet. bzw. feststellt, dass es nicht in A liegt.

0	1	2	3	4	5	6	7
3	5	7	11	15	21	28	33

Suchen: Aufgabe 3

Zeige schrittweise, wie die binäre Suche das Element 6 im Array A findet. bzw. feststellt, dass es nicht in A liegt.

0	1	2	3	4	5	6	7
3	5	7	11	15	21	28	33

Suchen: Aufgabe 4

Welche Voraussetzung muss eine Liste für die binäre Suche erfüllen?

Suchen: Aufgabe 5

Wie gross ist die Laufzeitkomplexität der linearen Suche nach einem Element x in einer Liste L im besten, im durchschnittlichen, und im schlechtesten Fall? Begründe die Antworten.

Hinweis: Das lineare Suche soll die Position des Elements oder `True`/`False` ausgeben, sobald es das Element x gefunden hat und danach stoppen.

Suchen: Aufgabe 6

Wie gross ist die Laufzeitkomplexität der binäre Suche nach einem Element x in einer Liste L der Länge n im Worst Case?

Bonusfrage: Wie sieht es im Best Case aus? [Das haben wir im Unterricht nicht thematisiert, lässt sich aber mit dem Wissen über den Algorithmus der binären Suche beantworten.]

Suchen: Aufgabe 7

Lohnt es sich, eine Liste extra zu sortieren, um eine binäre Suche darauf auszuführen. Begründe.

Suchen: Aufgabe 8

(a) Was ist *String-Matching*?

(b) Zähle mindestens drei Anwendungen des String-Matching auf.

Suchen: Aufgabe 9

Wie viele laufzeitrelevante Schritte benötigt die naive Suche, um das Muster `p = 1011` in in der Zeichenfolge `t = 1010010110` zu finden?

Suchen: Aufgabe 10

Gib asympototische Laufzeitkomplexität im Best Case und im Worst Case für die naive Suche nach einem Muster p der Länge m in einem Text t der Länge n an. Gib Beispiele für beide Fälle an.

Suchen: Aufgabe 11

Zeige möglichst ausführlich, wie der Algorithmus von Boyer-Moore-Horspool nach dem Suchmuster `GCTT` im Text `CCGTGGTAGCTT` sucht.

C	C	G	T	G	G	T	A	G	C	T	T

Suchen: Aufgabe 12

(a) Welcher der beiden String-Matching-Algorithmen ist schneller? Das Naive Verfahren oder der Algorithmus von Boyer, Moore und Horspool?

(b) Interpretiere die Daten in der folgende Tabelle:

Gesamtdauer für die Suche aller Vorkommen von 100 zufälligen Mustern p der Länge $|p|$ in einem zufälligen Text t der Länge $|t|$ über dem Alphabet $\Sigma = \{\texttt{01}\}$.

$\|p\|$	$\|t\|$	Naiv	BMH
2	10000	0.2240s	0.2714s
3	10000	0.2480s	0.2498s
4	10000	0.2612s	0.2480s
5	10000	0.2654s	0.2569s
6	10000	0.2679s	0.2605s
7	10000	0.2648s	0.2530s

TSP: Aufgabe 1

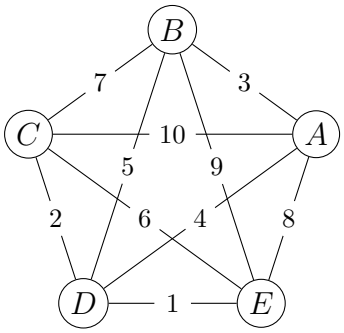
Nenne zwei (oder mehr) Anwendungen wirklich verschiedene Anwendungen für das Travelling Salesman Problem.

TSP: Aufgabe 2

Wie viele verschiedene Rundreisen gibt es bei 5 Städten, wenn wir in der ersten Stadt (nennen wir sie A) beginnen?

TSP: Aufgabe 3

Bestimme die Distanzmatrix zum folgenden kantengewichteten Graphen



TSP: Aufgabe 4

Schreibe eine Python-Funktion `factorial(n)`, die den Wert von $n!$ zurückgibt.

TSP: Aufgabe 5

(a) Skizziere eine Darstellung des gerichten Graphen G, der duch den folgenden Python-Code definiert wird. *Hinweis:* Der Graph ist nicht vollständig.

```
G = {
    'A': ['B', 'D'],
    'B': ['C', 'E'],
    'C': ['A', 'B', 'E'],
    'D': ['C'],
    'E': []
}
```

(b) Gibt es in G Zyklen? Wenn ja, beschreibe *einen* durch Angabe der Knoten, die auf ihm liegen.

TSP: Aufgabe 6

Wie viele Knoten hat ein vollständiger ungerichteter Graph mit 78 Kanten?

TSP: Aufgabe 7

Leite die Laufzeitkomplexität für die Brute Force-Lösung des TSPs mit n Städten her.

TSP: Aufgabe 8

Bestimme für die folgende Distanzmatrix die Länge der kürzesten Tour(en) des zugehörigen Travelling Salesman Problems. Es genügt, eine optimale Tour anzugeben.

	A	B	C	D
A	0	1	2	2
B	1	0	6	3
C	2	6	0	11
D	2	3	11	0

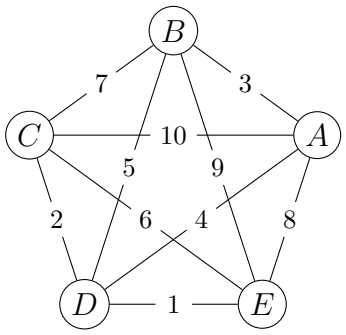
TSP: Aufgabe 9

Eine Implementierung des Brute Force-Algorithmus zur Lösung des Travelling Salesman Problems (TSP) für 14 Städte benötigt auf einem Computer etwa 2 Stunden.

Wie lange wird dieselbe Implementierung auf demselben Computer zur Lösung eines TSPs mit 16 Städten ungefähr benötigen? Falls die Lösung länger als 24 Stunden bzw. eine Woche dauert, stelle das Resultat in Wochen, Tagen und Stunden dar.

TSP: Aufgabe 10

(a) Bestimme die Länge der Rundtour für den den folgenden Graphen mit der Nearest-Neighbor-Heuristik und dem Startknoten A. Gibt es eine kürzere Route? Wenn ja, gib eine (ohne ausführlichen Lösungsweg) an.



(b) Gib jeweils einen Vor- und einen Nachteil der Nearest-Neighbor-Heuristik gegenüber der Brute-Force-Methode bei der Lösung des TSPs an.