Übungen zu Algorithmen
WSI für Informatik
Kaufmann/Kuckuk/Förster

Wintersemester 2021/22
Universität Tübingen
21. Dezember 2021

Übungsblatt 9

Abgabe bis 22.12.2021, 8:00 Uhr

Besprechung: 10.01.2021, 16:15 Uhr

Übungspartner: Rune Schwarz

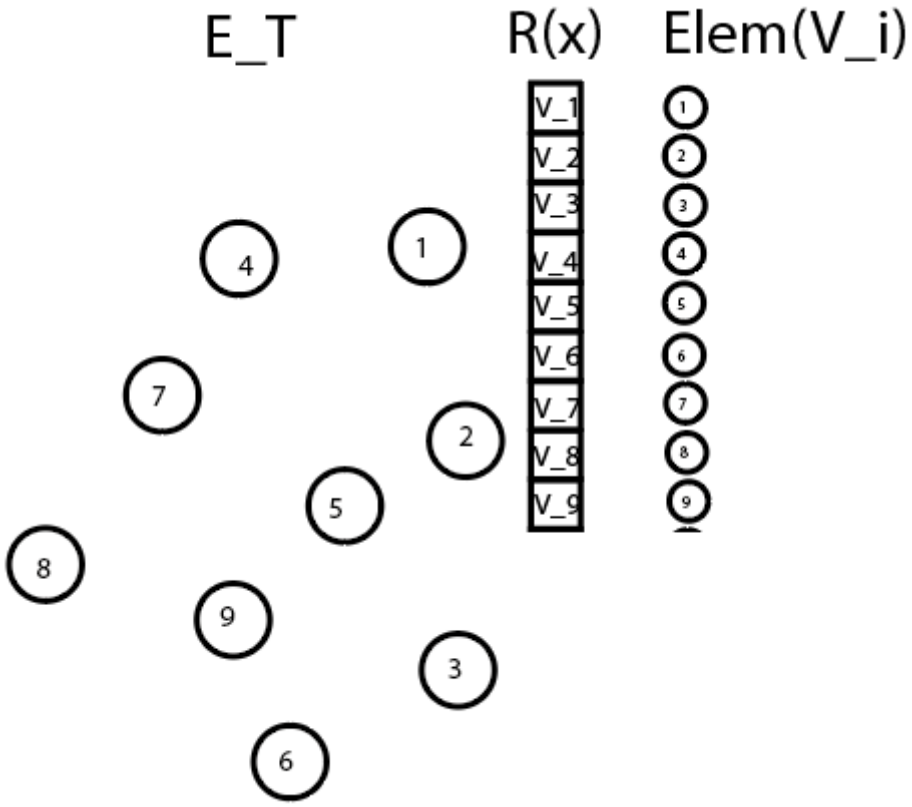
Aufgabe 1: Union-Find (5 Punkte)

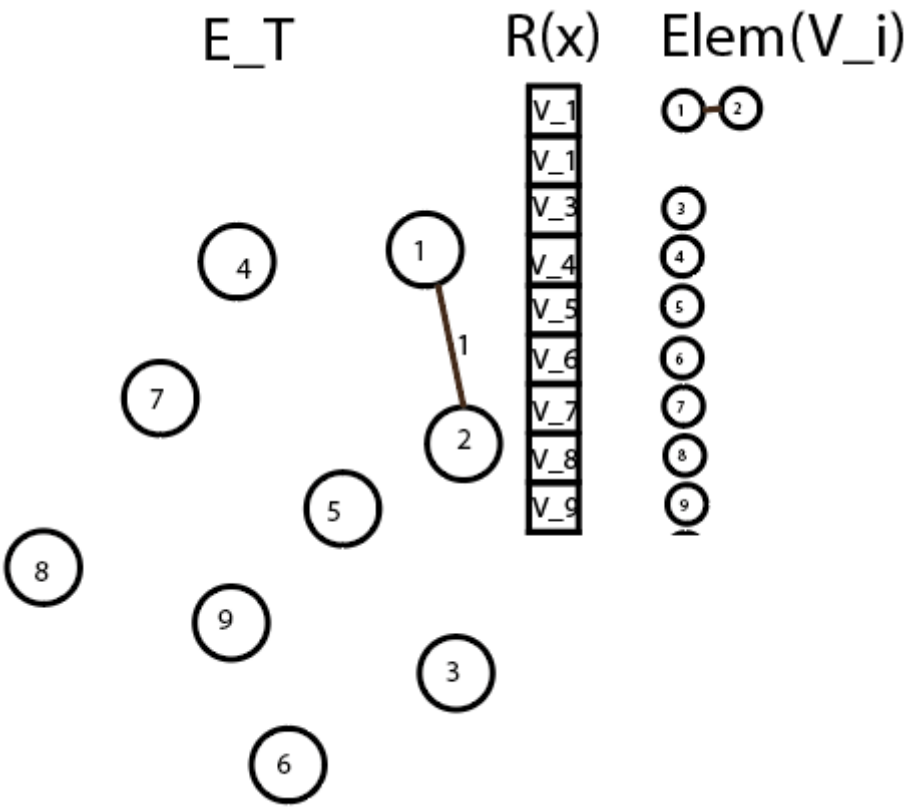
★ ★ ★ ☆ ☆

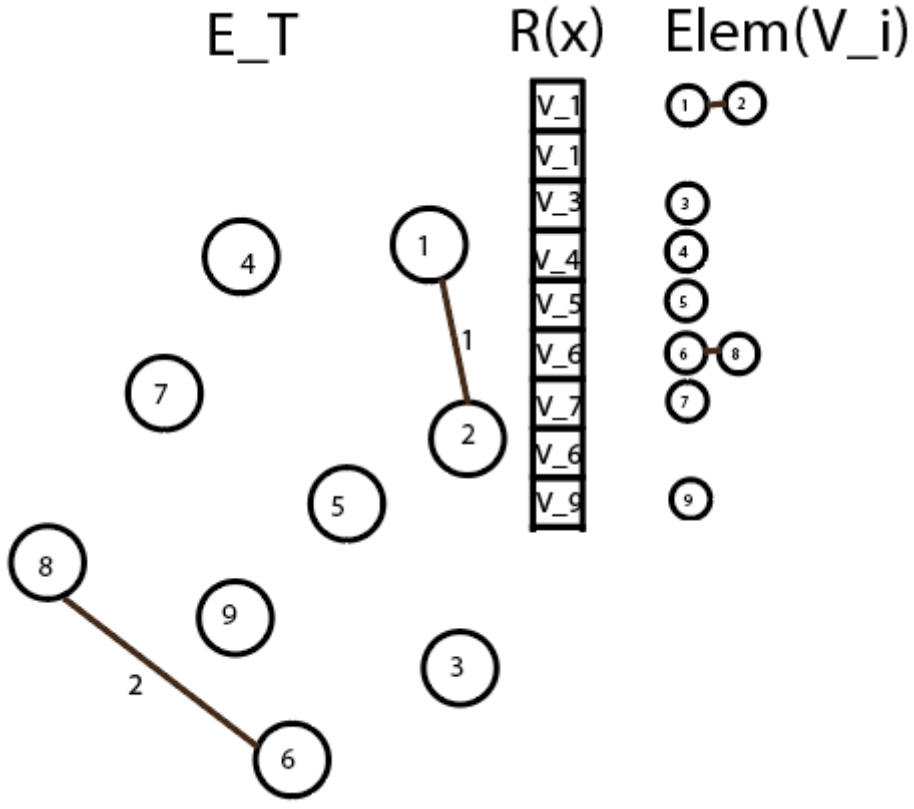
Um mit den gestiegenen Geschenkeanfragen mithalten zu können hat sich der Weihnachtsmann entschieden die einzelnen Stationen des Nordpols mit einem Rohrpostsystem zu verbinden. Dabei soll am Ende eine Verbindung über Rohrpoststrecken zwischen jedem Paar an Stationen existieren. Weiterhin sollen das Rohrpostsystem so angelegt werden, dass eine möglichst kurze Strecke von bereits vorhandenen Wegen zu einer Rohrpoststrecke ausgebaut werden muss.
Der Nordpol ist wie folgt angelegt. Die Knoten entsprechen den Stationen, die Kanten den bereits vorhandenen Wege, welche ausgebaut werden können. Die Länge eines Weges steht neben der entsprechenden Kante.

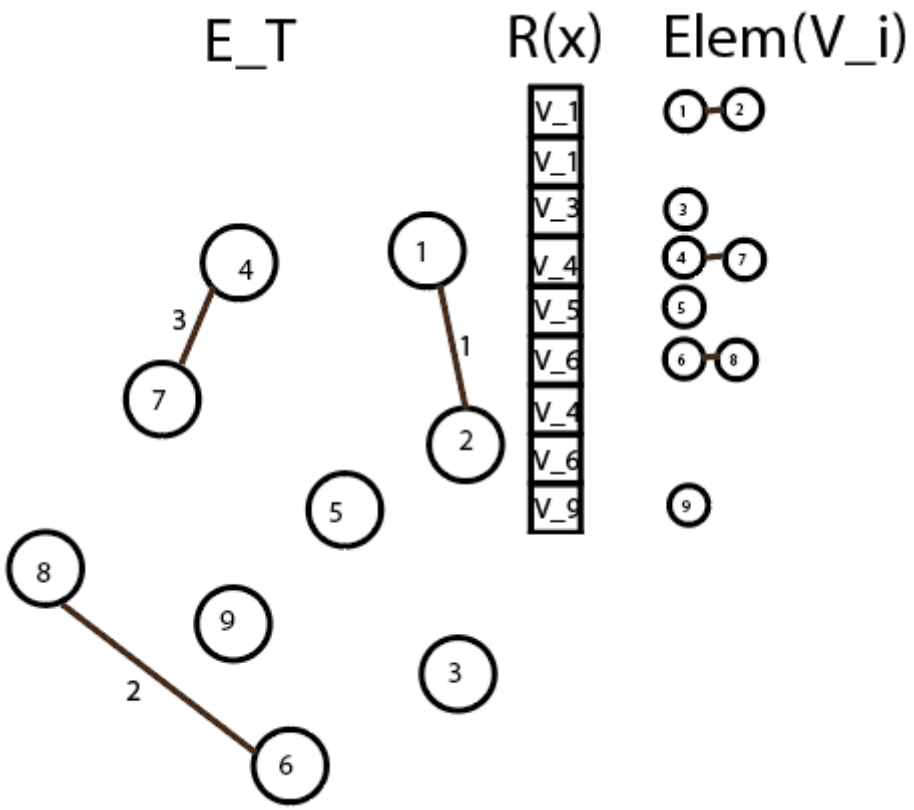
Nutzen Sie den Kruskal-Algorithmus um den Verlauf des Rohrpostsystems zu bestimmen. Geben Sie dabei E_T nach jedem Schritt an. Geben Sie außerdem in jedem Schritt den aktuellen Zustand der Union-Find Datenstruktur an, genauer: das Namensfeld R und die Liste 'Elem(A)' für jede Menge A.
Dabei können Sie die Knoten mit der angegebenen Ziffer anstatt dem Stationsnamen bezeichnen.

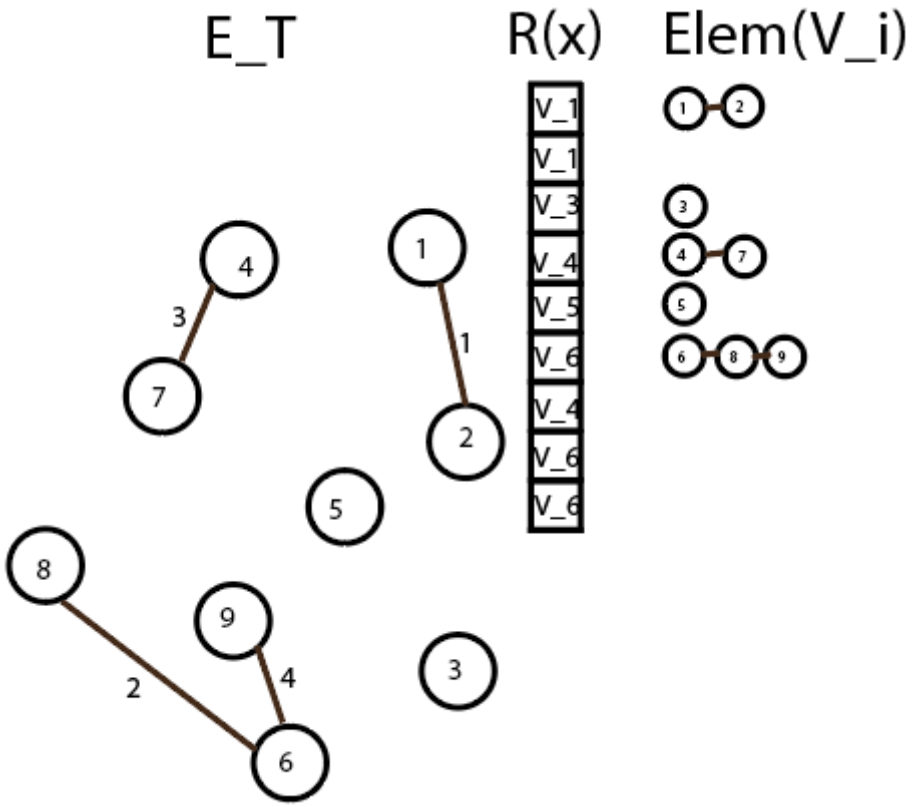
1 Lösung Aufgabe 1

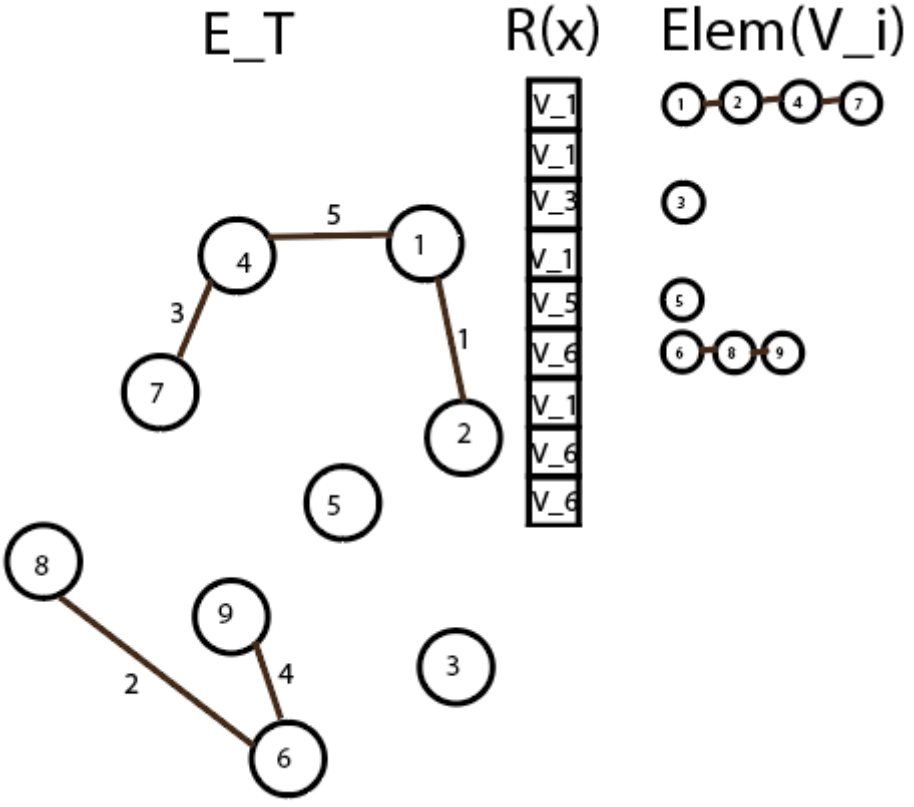


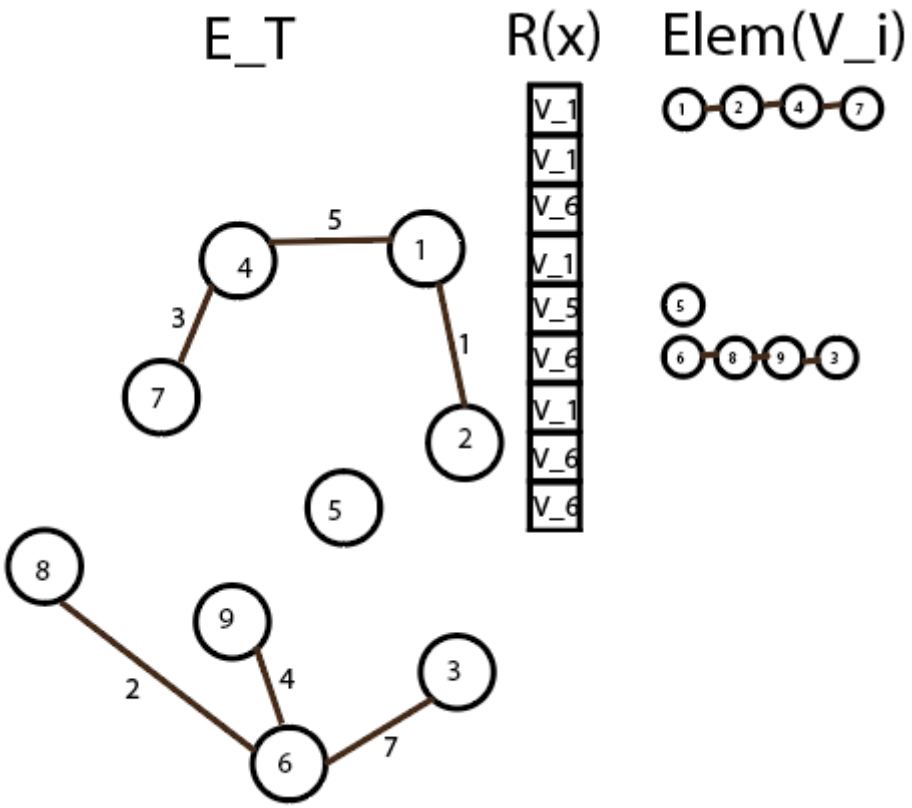

E_T
R(x)
Elem(V_i)
V_1
V_1
V_3
V_4
V_5
V_6
V_7
V_8
V_9
1
2
3
4
5
6
7
8
9
1

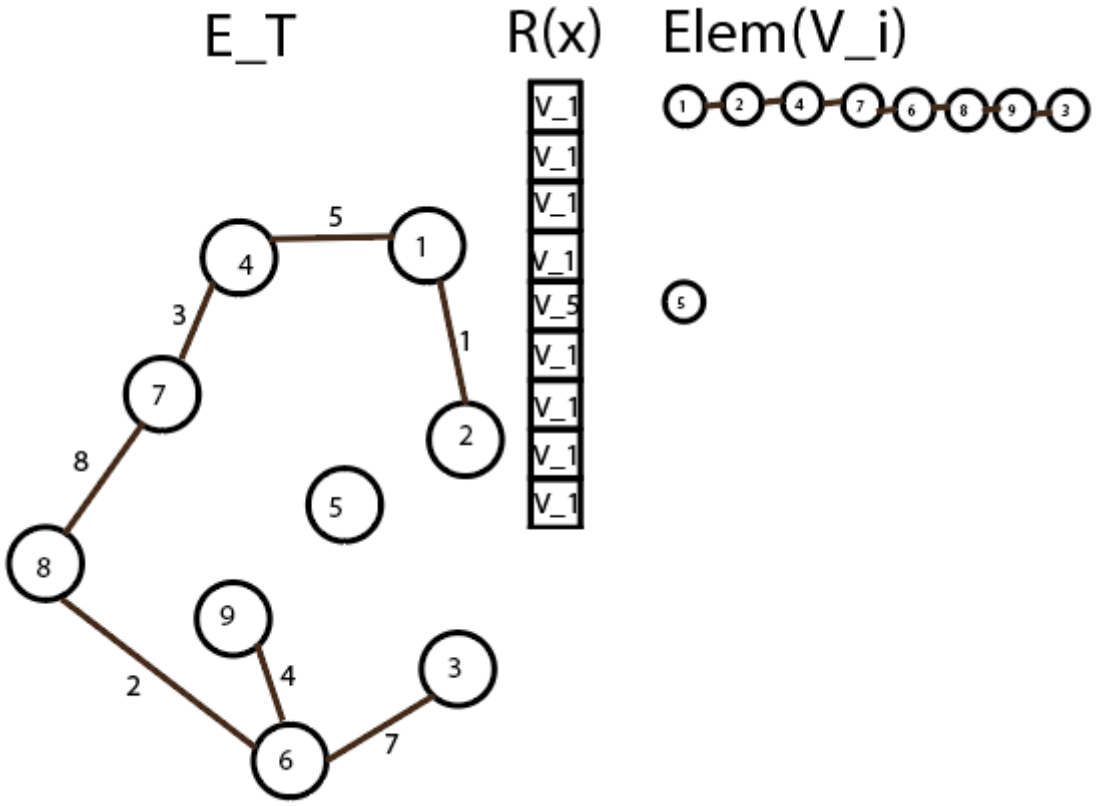

E_T
R(x)
Elem(V_i)
V_1
V_1
V_3
V_4
V_5
V_6
V_7
V_6
V_9
1
2
3
4
5
6
7
8
9
1
2

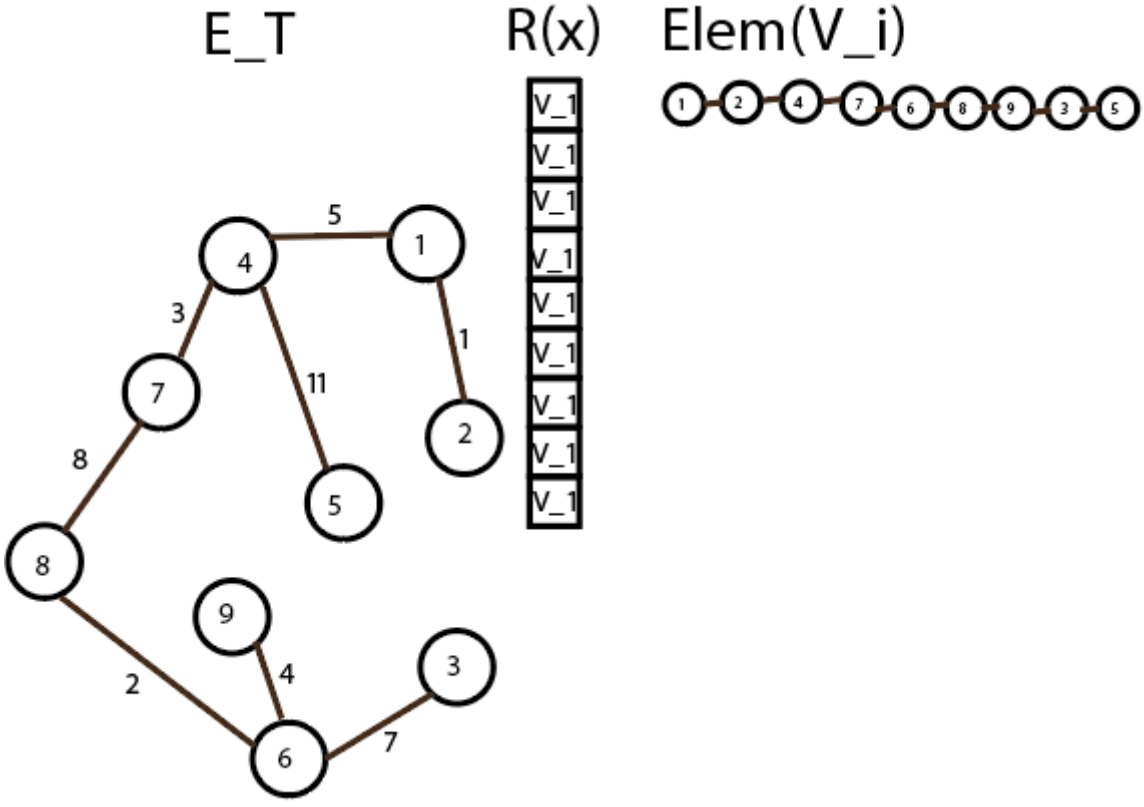

E_T
R(x)
Elem(V_i)
V_1
V_1
V_3
V_4
V_5
V_6
V_4
V_6
V_9
1
2
3
4
5
6
7
8
9
1
3
2


E_T
R(x)
Elem(V_i)
V_1
V_1
V_3
V_4
V_5
V_6
V_4
V_6
V_6
1
2
3
4
5
6
7
8
9


E_T
R(x)
Elem(V_i)
V_1
V_1
V_3
V_1
V_5
V_6
V_1
V_6
V_6
1
2
3
4
5
6
7
8
9


E_T
R(x)
Elem(V_i)
V_1
V_1
V_6
V_1
V_5
V_6
V_1
V_6
V_6
1 2 4 7
5
6 8 9 3
5
4
1
7
2
8
9
3
6
5
3
1
2
4
7


E_T
R(x)
Elem(V_i)
V_1
V_1
V_1
V_1
V_5
V_1
V_1
V_1
V_1
1 2 4 7 6 8 9 3
5
5
1
4
2
3
7
8
9
6
1
3
4
5
7
2


E_T
R(x)
Elem(V_i)
V_1
V_1
V_1
V_1
V_1
V_1
V_1
V_1
V_1
1 2 4 7 6 8 9 3 5
5
1
4
2
3
7
8
9
6
11
1
3
4
5
7
2

Aufgabe 2: Verschiedene Suchstrategien (1+1+2+2 Punkte) ★★★★☆

Die Weihnachtsvorbereitungen sind bereits in vollem Gange, doch da stellt sich heraus, dass der Grinch sich in die Elfenwerkstatt geschlichen und am Fließband Weihnachtsgeschenke sabotiert hat. Natürlich wird das Fließband ´sofort gestoppt und der Grinch vertrieben.

Es verbleibt jedoch eine lange Reihe an n Geschenken auf dem Fließband wovon der Grinch die letzten k Geschenke (mit $0 \leq k \leq n$) manipuliert hat. Diese müssen nun natürlich identifiziert und neutralisiert werden, doch leider ist von außen nicht zu sehen, an welchen Päckchen gewerkelt wurde. Es müssen also Geschenke geöffnet werden um festzustellen ob diese sabotiert sind oder nicht. Dabei sollen Sie nun aushelfen.

Sie stehen dabei neben dem stillgelegten Fließband und die ältesten Päckchen befinden sich rechts, die neuen und vom Grinch sabotierten links. Sie können also annehmen, dass die Geschenke sich in einer Ordnung befinden, in welcher die $n-k$ sicheren (d.h. nicht vom Grinch manipulierten) Geschenke rechts und die k sabotierten auf der linken Seite stehen.

a) Geben Sie eine Strategie in Form eines Algorithmus an, mit welcher alle sabotierten Geschenke gefunden werden. Dabei sollen möglichst wenige Geschenke geöffnet werden (egal ob sabotiert oder nicht). Geben Sie die nötige Anzahl an geöffneten Geschenken an und begründen Sie ihre Antwort.

b) Geben Sie eine Strategie in Form eines Algorithmus an, mit welcher alle sabotierten Geschenke gefunden werden. Dabei sollen nun möglichst wenige sichere Pakete geöffnet werden, da nicht mehr viel Zeit bis zum Weihnachtsabend verbleibt, um die geöffneten Geschenke neu zu verpacken.

   Geben Sie die nötige Anzahl von geöffneten Geschenken insgesamt an, geben Sie an wie viele sichere Geschenke geöffnet wurden und begründen Sie ihre Antwort.

Schnell wird klar, dass der Grinch in den sabotierten Geschenken Springteufel versteckt hat. Nun sind Elfen allerdings so schreckhaft, dass sie sofort in Schock verfallen, wenn sie eines der sabotierten Päckchen öffnen und vorerst nicht weiterarbeiten können. Inzwischen stehen aber nur noch zwei tapfere Elfen zur Verfügung. Es kann also maximal zweimal ein sabotiertes Geschenk geöffnet werden um alle sabotierten Geschenke zu identifizieren.

Sicherlich könnten Sie sofort die Lösung von Aufgabenteil a) so abändern, dass die Elfen möglichst geschont bleiben, allerdings gilt es weiterhin möglichst wenige Geschenke zu öffnen, da Weihnachten direkt vor der Tür steht. Es stehen inzwischen nur noch 169 Geschenke auf dem Fließband, also $n = 169$.

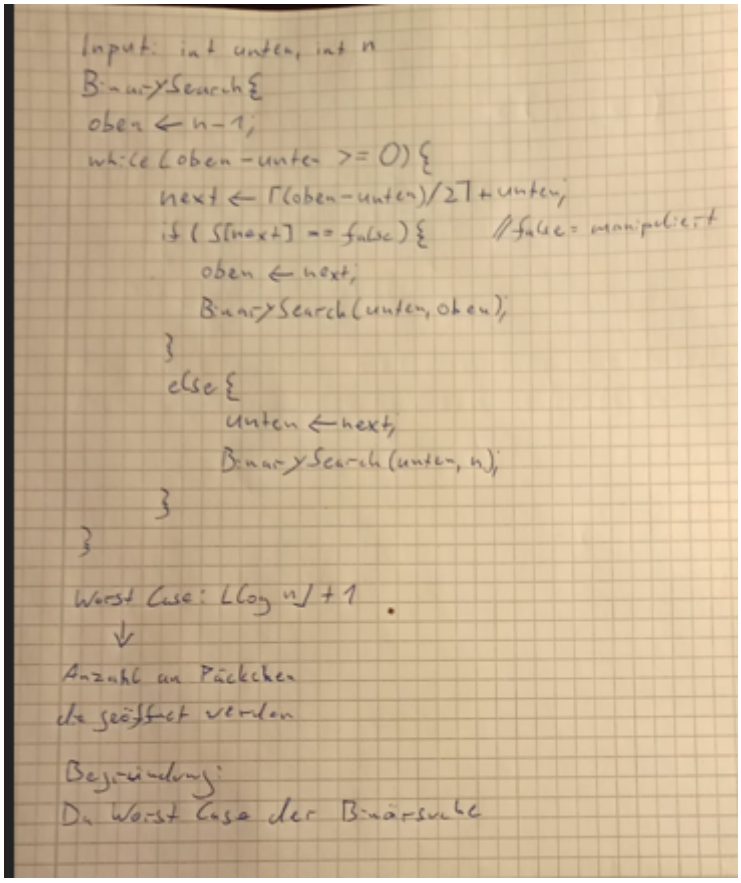
c) Zeigen Sie, dass die zwei Elfen mit höchstens 26 geöffneten Geschenken alle sabotierten Geschenke identifizieren können.

   Tipp: Es kann helfen sich erst zu überlegen wie 84 geöffnete Pakete ausreichen um die sabotierten zu identifizieren.

d) Wenn Sie ihren Algorithmus von Aufgabenteil c) optimieren, wie viele geöffnete Pakete sind minimal nötig? Begründen Sie ihre Antwort.

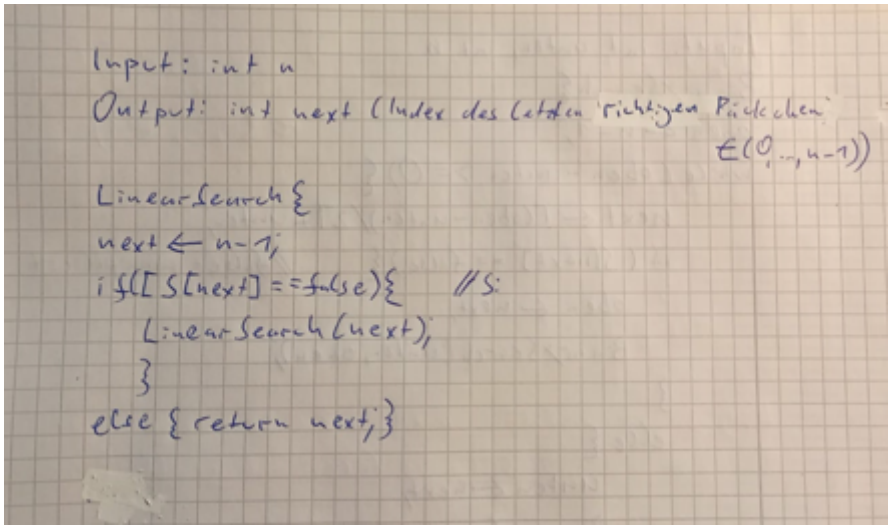
2 Lösung Aufgabe 2

a)



S: Array in dem gesucht wird.

b)



Insgesamt wurdne k+1 Geschenke geöffnet. Davon wurde 1 sicheres Geschenk geöffnet.

S: Array, in dem gesucht wird.

Aufgabe 3: Interpolationssuche (4 Punkte) ★★★☆☆

Der Weihnachtsmann ist sehr zufrieden mit dem neuen Rohrpostsystem und möchte nun auch andere am Nordpol anfallende Aufgaben automatisieren. Als nächstes steht die Liste an, auf der der Weihnachtsmann notiert, welches Kind lieb und welches unartig gewesen ist.

Das zu entwickelnde Programm soll bei einem Kind Die Liste liegt bereits als `.csv`-Datei vor das zu entwickelnde Programm soll bei einem gegebenen Namen eines Kindes vermerken, ob dieses `NAUGHTY` oder `NICE` war. Mit den bereits implementierten Funktionen in `NaughtyOrNice.cpp` kann die Liste gelesen und auch wieder gespeichert werden. Allerdings fehlt nun noch die Implementierung einer Funktion, die effizient den Name eines bestimmten Kindes in der Liste suchen kann.

Aufgabe: Implementieren Sie in der Datei `InterpolationSearch.cpp` die Funktion `int interpolationSearch(string item, vector<string> list, int listLength)`. Die Funktion soll mittels Interpolationssuche (die verbesserte Version mit Laufzeit $\mathcal{O}(\sqrt{n})$ wie in der VL und im Pseudocode auf dem Präsenzübungsblatt beschrieben) den `string item` innerhalb des sortierten `vector<string> list` der Länge `listLength` suchen und den Index von `item` zurückgeben. Ist `item` nicht in `list` enthalten, soll stattdessen -1 zurückgegeben werden. Hier noch einige Hinweise:

- Auf die Elemente eines `vector` kann wie auf die Elemente eines Arrays zugegriffen werden, d.h. `list[i]` gibt das i-te Element von `list` zurück.
- Für die Strings `a` und `b` ist der Wert von `a.compare(b)` genau dann
  - (i) 0, wenn `a` und `b` identisch sind,
  - (ii) `<` 0, wenn `a` vor `b` in einer alphabetischen Sortierung auftaucht, bzw.
  - (iii) `>` 0, wenn `a` nach `b` in einer alphabetischen Sortierung auftaucht.

  Die Liste `list` ist alphabetisch sortiert.
- Die bereits gegebene Funktion `double stringQuotient(string a, string b, string c)` berechnet ein Maß für den Quotienten $\frac{b-a}{c-a}$ sofern `a` alphabetisch vor `b` kommt und `b` alphabetisch vor `c`, ansonsten wird -1 zurückgegeben. Nutzen Sie diese Funktion für die Berechnung von $\frac{x-A[unten]}{A[oben]-A[unten]}$!
- In `NaughtyOrNice.cpp` können die Zeilen 144 bis 155 entkommentiert werden, um zu überprüfen, ob Ihre Implementierung alle Namen finden kann. Wenn Sie ihrem Programmaufruf einen nicht in der Liste enthaltenen Namen übergeben, können Sie auch überprüfen, dass dies korrekt identifiziert wird.
- Beachten Sie, dass `list.csv` im Verzeichnis, in dem Ihr Programm ausgeführt wird, enthalten ist. Alternativ können Sie in Zeile 68 von `NaughtyOrNice.cpp` die Variable `inFileName` auch den Pfad von `list.csv` setzen.

Bitte geben Sie Ihre Lösung als Quelltext (.cpp/.h) ab. Bitte komprimieren Sie alle Dateien (inkl. `list.csv`), die zum Kompilieren nötig sind und laden diese als Zip-Archiv hoch. Ist Ihr Code nicht kompilierbar oder treten Exceptions bei dem in der Main-Funktion gegebenen Beispiel auf, so wird ihr Code nicht bewertet.