Aufgabe 3: Wandertag

Team-ID: 00153

Team-Name: ByteBusters

Bearbeiter/-innen dieser Aufgabe:
Benedikt Wiesner

17. November 2024

Inhaltsverzeichnis

---

1 Lösungsidee

Die Kernidee der Lösung basiert auf einem Greedy-Algorithmus, der darauf abzielt, die maximale Anzahl von Teilnehmern mit drei optimal gewählten Streckenlängen abzudecken. Der Algorithmus wird in den folgenden Schritten ausgeführt:

1. **Erstellung der Streckenlängenliste**: Zunächst werden alle möglichen Streckenlängen ermittelt, indem die minimalen und maximalen Streckenlängen aller Teilnehmer gesammelt werden. Diese Streckenlängen stellen die potenziellen Werte dar, die für die Abdeckung der Teilnehmer erforderlich sind.

2. **Berechnung der Abdeckung für jede Streckenlänge**: Für jede dieser möglichen Streckenlängen wird berechnet, wie viele neue Teilnehmer sie abdeckt, die bisher noch nicht durch eine der bereits gewählten Streckenlängen erfasst wurden. Diese Berechnung erfolgt unter Berücksichtigung der individuellen Anforderungen der Teilnehmer.

3. **Auswahl der optimalen Streckenlänge**: Die Streckenlänge, die die größte Anzahl neuer Teilnehmer abdeckt, wird ausgewählt und zur aktuellen Lösung hinzugefügt. Dies stellt sicher, dass bei jeder Wahl die Abdeckung maximiert wird.

4. **Wiederholung des Auswahlprozesses**: Dieser Prozess wird wiederholt, bis insgesamt drei Streckenlängen ausgewählt wurden oder bis keine weiteren Teilnehmer mehr durch zusätzliche Streckenlängen abgedeckt werden können. Sollte es keine weiteren Teilnehmer geben, die durch eine zusätzliche Streckenlänge erfasst werden können, endet der Algorithmus vorzeitig.

Dieser Greedy-Ansatz maximiert in jeder Runde die Anzahl der neu abgedeckten Teilnehmer, was zu einer optimalen oder nahezu optimalen Lösung führt. Der Algorithmus trifft dabei lokale Entscheidungen, ohne langfristige Auswirkungen zu berücksichtigen. Aufgrund der begrenzten Anzahl an Streckenlängen wird er in akzeptabler Zeit ausgeführt und liefert in vielen Fällen eine Lösung, die der optimalen sehr nahekommt.

---

2 Umsetzung

Die Implementierung der Lösung wurde in Python realisiert und umfasst mehrere Hauptkomponenten, die in den folgenden Untersektionen beschrieben werden.

2.1 Dateneinlese

Zu Beginn des Programms werden die Eingabedaten aus einer Textdatei eingelesen, die im gleichen Verzeichnis wie das Skript abgelegt ist. Die erste Zeile enthält die Anzahl der Personen, gefolgt von den minimalen und maximalen Streckenlängen jeder Person.

2.2 Greedy-Algorithmus

Der Greedy-Algorithmus bildet das zentrale Element der Lösung. Ziel des Algorithmus ist es, mit der Auswahl von genau drei Streckenlängen die maximale Anzahl an Teilnehmern abzudecken. In jeder Iteration wird die Streckenlänge ausgewählt, die die größte Anzahl an neuen, bisher nicht abgedeckten Teilnehmern erreicht.

Formell betrachtet:

- Gegeben sei eine Menge von Teilnehmern $P = \{p_1, p_2, \ldots, p_n\}$, wobei jeder Teilnehmer p_i mit einer minimalen und einer maximalen Streckenlänge $[\min_i, \max_i]$ verbunden ist.
- Zu Beginn sind noch keine Streckenlängen gewählt und keine Teilnehmer abgedeckt. Das Ziel des Algorithmus ist es, mit jeder neuen Streckenlänge die Anzahl der abgedeckten Teilnehmer zu maximieren.

Der Algorithmus wird in mehreren Schritten durchgeführt:

1. **Initialisierung**: Zu Beginn wird eine Liste aller möglichen Streckenlängen erstellt. Diese Streckenlängen können als alle Werte im Bereich von $\min_i$ bis $\max_i$ für jeden Teilnehmer betrachtet werden. Jede dieser Streckenlängen bildet die Grundlage für die Auswahl.
2. **Iterative Auswahl**: In jeder Iteration wird die Streckenlänge ausgewählt, die die größte Anzahl an neuen, noch nicht abgedeckten Teilnehmern erreicht. Eine Streckenlänge s deckt genau diejenigen Teilnehmer ab, deren minimaler Wert $\min_i$ kleiner oder gleich s und deren maximaler Wert $\max_i$ größer oder gleich s ist. Die Menge der neuen Teilnehmer für eine Streckenlänge s lässt sich formal wie folgt beschreiben:

   $$\text{Neue Teilnehmer}(s) = \{p_i \mid \min_i \leq s \leq \max_i\} \setminus \text{abgedeckte Teilnehmer}.$$

   Die Streckenlänge s_{best}, die die größte Anzahl an neuen Teilnehmern abdeckt, wird durch Maximierung der Anzahl der neuen Teilnehmer ausgewählt:

   $$s_{\text{best}} = \arg\max_s |\text{Neue Teilnehmer}(s)|.$$

3. **Abbruchbedingung**: Dieser Prozess wird wiederholt, bis entweder drei Streckenlängen ausgewählt wurden oder keine weiteren Teilnehmer mehr abgedeckt werden können.

Laufzeitanalyse: Die Laufzeit des Algorithmus hängt von der Anzahl der Streckenlängen und der Anzahl der Teilnehmer ab. Zu Beginn müssen alle möglichen Streckenlängen aus den minimalen und maximalen Werten der Teilnehmer erstellt werden. Dies bedeutet, dass für jeden Teilnehmer der Bereich von $\min_i$ bis $\max_i$ alle möglichen Werte betrachtet werden.

Für jede Streckenlänge müssen wir dann die Anzahl der abgedeckten Teilnehmer berechnen. Dies erfordert das Durchlaufen der Teilnehmerliste, was eine Zeitkomplexität von $O(n)$ pro Streckenlänge bedeutet.

Da der Algorithmus maximal drei Streckenlängen auswählt, ergibt sich insgesamt folgende Laufzeit:

- **Erstellung der Liste möglicher Streckenlängen:** Da jeder Teilnehmer mindestens eine minimale und eine maximale Streckenlänge hat, müssen alle Werte im Bereich $[\min_i, \max_i]$ als potenzielle Streckenlängen betrachtet werden. Dies führt zu $O(n \cdot \Delta)$, wobei Δ der Bereich (d.h. der Unterschied zwischen dem minimalen und maximalen Wert) ist.

- **Berechnung der neuen Teilnehmer für jede Streckenlänge:** Es wird für jede Streckenlänge überprüft, welche Teilnehmer durch diese abgedeckt werden. Dies hat eine Zeitkomplexität von $O(n)$ pro Streckenlänge.
- **Auswahl der besten Streckenlänge (maximal 3 Iterationen):** Insgesamt wird dieser Schritt mit einer maximalen Anzahl von drei Streckenlängen wiederholt. Daher ergibt sich eine Gesamtlaufzeit von $O(3 \cdot n \cdot \Delta)$, also $O(n \cdot \Delta)$.

Die Gesamtlaufzeit des Algorithmus beträgt daher $O(n \cdot \Delta)$, was eine lineare Laufzeit in Bezug auf die Anzahl der Teilnehmer und den Streckenbereich darstellt.

```python
def greedy_algorithm(personen):
    gewählte_strecken = []
    abgedeckte_personen = set()

    strecken_kandidaten = set()
    for min_strecke, max_strecke in personen:
        # Füge alle Strecken zwischen der minimalen und maximalen Strecke hinzu
        for strecke in range(min_strecke, max_strecke + 1):
            strecken_kandidaten.add(strecke)


    for _ in range(3):
        if not strecken_kandidaten:
            break

        beste_strecke = None
        beste_abdeckung = set()


        for strecke in strecken_kandidaten:
            aktuelle_abdeckung = set(teilnehmer_zahlen(strecke, personen))
            neue_abdeckung = aktuelle_abdeckung - abgedeckte_personen

            if len(neue_abdeckung) > len(beste_abdeckung):
                beste_strecke = strecke
                beste_abdeckung = neue_abdeckung

        if beste_strecke is None:
            break

        gewählte_strecken.append(beste_strecke)
        abgedeckte_personen.update(beste_abdeckung)
        strecken_kandidaten.discard(beste_strecke)

    return gewählte_strecken, abgedeckte_personen
```

2.3 Hilfsfunktionen

Zur Unterstützung des Algorithmus wurden mehrere Hilfsfunktionen implementiert:

- `teilnehmer_zahlen`: Bestimmt, welche Teilnehmer eine bestimmte Strecke absolvieren können.
- `option_1`: Verarbeitet die Auswahl der zu scannenden Datei und führt den Greedy-Algorithmus aus.
- `Laufzeitmessung`: Misst die Laufzeit des gesamten Programms, indem die Zeit vor und nach der Algorithmus-Ausführung erfasst und die Differenz berechnet wird.

Mit diesen Komponenten wurde eine effiziente und übersichtliche Lösung für das Problem implementiert.

---

3 Beispiele

Das Format der Datei besteht aus der Anzahl der Personen gefolgt von den minimalen und maximalen Zeiten in Minuten für jede Person. Jede Zeile enthält für eine Person zwei Werte: die minimale und die maximale Zeit.

$$\text{Anzahl der Personen} = 7$$
$$\begin{pmatrix} 12 & 35 \\ 22 & 45 \\ 46 & 46 \\ 48 & 62 \\ 51 & 57 \\ 64 & 64 \\ 64 & 71 \end{pmatrix}$$

Da der Index bei 0 beginnt, wird die erste Person als Person 0 ausgegeben.

Dateiname	**Ausgabe (vereinfacht)**	**Laufzeit**
wandern1.txt //kleine Datenmenge	• Gewählte Streckenlängen: [22, 51, 64] • Teilnehmende Personen: • Strecke 22: [0, 1] • Strecke 51: [3, 4] • Strecke 64: [5, 6]	0.0030 s
wandern7.txt //große Datenmenge	• Gewählte Streckenlängen: [52515, 86493, 30966] • Strecke 52515,6493,30966: [Ausgabe zu groß]	17.0751 s
leer.txt //leere Datei	• Die Datei ist leer.	N/A
eine.txt // nur eine Eingabe	• Gewählte Streckenlängen: [50] • Teilnehmende Personen: [0]	0.0019 s
gleich.txt //gleiche Eingaben	• Gewählte Streckenlängen: [42] • Teilnehmende Personen: [0, 1, 2, 3, 4]	0.0020 s
keine.txt //keine Überschneidungen	• Gewählte Streckenlängen: [70, 40, 10] • Teilnehmende Personen: [0, 1, 3] • Strecke 10: [0] • Strecke 30: [1] • Strecke 50: [2]	0.0030 s

Tabelle 1: Ergebnisse der Dateiverarbeitung (alle BWINF Beispieleingaben wurden erfolgreich getestet)

4 Diskussion und Ausblick

Der Greedy-Algorithmus liefert in den meisten Fällen eine optimale oder nahezu optimale Lösung. Er ist effizient und einfach zu implementieren. Allerdings gibt es einige Punkte zu beachten:

- Der Algorithmus garantiert nicht immer die global optimale Lösung, da er in jedem Schritt die lokal beste Wahl trifft.
- In Extremfällen, wo eine globale Betrachtung notwendig wäre, könnte der Algorithmus suboptimale Ergebnisse liefern.

Für zukünftige Verbesserungen könnten wir folgende Ansätze in Betracht ziehen:

- Implementierung eines exakten Algorithmus (z.B. dynamische Programmierung) für kleinere Datensätze, um die Optimalität zu garantieren.
- Entwicklung eines hybriden Ansatzes, der den Greedy-Algorithmus mit lokaler Suche oder Metaheuristiken kombiniert, um die Lösungsqualität zu verbessern.
- Parallelisierung des Algorithmus für größere Datensätze, um die Laufzeit zu reduzieren.

5 Quellcode

Der Quellcode ist in mehrere Abschnitte unterteilt, um die einzelnen Schritte des Algorithmus klar darzustellen:

5.1 Greedy-Algorithmus

```
1
def greedy_algorithm(personen):
    gewählte_strecken = []
    abgedeckte_personen = set()
    strecken_kandidaten = set()
    for min_strecke, max_strecke in personen:
        for strecke in range(min_strecke, max_strecke + 1):
            strecken_kandidaten.add(strecke)
    for _ in range(3):
        if not strecken_kandidaten:
            break
        beste_strecke = None
        beste_abdeckung = set()
        for strecke in strecken_kandidaten:
            aktuelle_abdeckung = set(teilnehmer_zahlen(strecke, personen))
            neue_abdeckung = aktuelle_abdeckung - abgedeckte_personen
            if len(neue_abdeckung) > len(beste_abdeckung):
                beste_strecke = strecke
                beste_abdeckung = neue_abdeckung
        if beste_strecke is None:
            break
        gewählte_strecken.append(beste_strecke)
        abgedeckte_personen.update(beste_abdeckung)
        strecken_kandidaten.discard(beste_strecke)
    return gewählte_strecken, abgedeckte_personen
```

5.2 Teilnehmerzahlen

```
def teilnehmer_zahlen(strecke, personen):
    return [i for i, (min_strecke, max_strecke) in enumerate(personen) if min_strecke <=
    strecke <= max_strecke]
```

5.3 Dateiauswahl und Verarbeitung

```
def option_1():
    script_dir = os.path.dirname(os.path.abspath(__file__))
    dateien = [f for f in os.listdir(script_dir) if f.endswith('.txt')]

    if not dateien:
        print("Keine Textdateien im Verzeichnis gefunden.")
        input("\nDrücken Sie Enter, um zum Hauptmenü zurückzukehren...")
        return

    print("Verfügbare Textdateien:")
    for idx, datei in enumerate(dateien, 1):
        print(f"{idx}. {datei}")

    try:
        auswahl = int(input("\nGeben Sie die Nummer der Datei ein, die verarbeitet werden soll: "))
        if 1 <= auswahl <= len(dateien):
            dateipfad = os.path.join(script_dir, dateien[auswahl - 1])
            start_zeit = time.time()
            with open(dateipfad, 'r') as datei:
                zeilen = datei.readlines()
                if len(zeilen) == 0:
                    print("Die Datei ist leer.")
                    input("\nDrücken Sie Enter, um zum Hauptmenü zurückzukehren...")
                    return
                eintragsanzahl = int(zeilen[0].strip())
                if eintragsanzahl == 0:
                    print("Die Datei enthält keine Einträge.")
                    input("\nDrücken Sie Enter, um zum Hauptmenü zurückzukehren...")
                    return
                personen = [tuple(map(int, zeile.strip().split())) for zeile in zeilen[1: eintragsanzahl + 1]]
            print("\nVerarbeite Datei:", dateipfad)
            strecken, teilnehmer = greedy_algorithm(personen)
            print("Gewählte Streckenlängen:", strecken)
            print("Teilnehmende Personen:", list(teilnehmer))
            for strecke in strecken:
                min_strecke = strecke
                max_strecke = strecke
                teilnehmer_bei_strecke = teilnehmer_zahlen(strecke, personen)
                print(f"Strecke {strecke}: Min: {min_strecke}, Max: {max_strecke}, Teilnehmende Personen: {teilnehmer_bei_strecke}")
            end_zeit = time.time()
            laufzeit = end_zeit - start_zeit
            print(f"\nLaufzeit für die Verarbeitung dieser Datei: {laufzeit:.4f} Sekunden")
        else:
            print("Ungültige Auswahl.")
    except ValueError:
        print("Ungültige Eingabe.")
    input("\nDrücken Sie Enter, um zum Hauptmenü zurückzukehren...")
```