Aufgabe 5: Wichteln

Team-ID: 00383

Team: Theodor Teslia

Bearbeiter/-innen dieser Aufgabe:
Theodor Teslia

22. November 2020

Inhaltsverzeichnis

Lösungsidee

Um das Problem zu lösen, werden zuerst drei Listen mit allen möglichen, bzw. sinnvollen Wünschen erstellt. In die erste Liste werden alle ersten Wünsche gespeichert, aber nur einzeln, so dass nichts doppelt vorkommt. In der zweiten Liste werden alle zweiten Wünsche gespeichert, die nicht in der ersten vorkommen und deren erster Wunsch (bzw. der erste Wunsch der Person, die diesen zweiten Wunsch besitzt) mehr als ein Mal bei den ersten Wünschen vorkommt. Die dritte Liste wird dann mit allen dritten Wünschen gefüllt, die nicht in den anderen beiden vorkommen und deren erster Wunsch mehr als ein Mal bei den ersten Wünschen vorkommt.

Darauf basierend wird dann immer eine Teilgruppe betrachtet, welche so sortiert ist, dass eine Teilgruppe nur aus Personen mit dem gleichen Erstwunsch besteht. Ab da besteht der Algorithmus nur noch aus drei Schritten. Im 1. wird der nächste Spieler, welcher ein Geschenk zugewiesen bekommen soll, ermittelt. Dies basiert darauf, wie viele Wünsche er erfüllt bekommen kann, wodurch der nächste Spieler der ist, der die wenigsten Möglichkeiten hat. Der nächste Teil des Algorithmus besteht darin, dem nächsten Spieler ein Geschenk zuzuordnen. Das funktioniert so, dass der Computer durch alle möglichen Geschenke iteriert und den besten findet (sei es der 1., 2. oder 3. Wunsch). Wenn dies für jeden Spieler gemacht wurde, werden alle Spieler, denen bisher kein Geschenk zugeordnet werden konnte, mit irgendeinem Geschenk versorgt, so dass am Ende jeder ein Geschenk besitzt.

Umsetzung

Auch hier war die benutzte Sprache C++. Die Implementation der Liste zum Darstellen aller möglichen Wünsche habe ich durch folgende Struktur realisiert: `std::array<std::unordered_set<int>, 3>`. Wobei das `set` am Index 0 die Liste der ersten Wünsche darstellt usw. Dadurch hatte ich die Möglichkeit STL-Methoden, wie `std::unordererd_set::find()` oder `std::map::at()` zu benutzen.

Die Methode, um den nächsten Spieler zu ermitteln, habe ich so geschrieben, dass man einen normalen, linearen Suchalgorithmus zum Suchen der kleinsten Zahl benutzt, wobei aber auch darauf geachtet wird, dass das neue Minimum noch nicht zugeordnet wurde und ob es überhaupt eine Anzahl an erfüllbaren Wünschen besitzt. Der dadurch resultierende Spieler ist dann der nächste.

Um herauszufinden, welcher Wunsch denn benutzt werden sollte, wird zuerst überprüft, ob der zweite Wunsch der ausgewählten Person einzigartig ist, er also direkt zugeordnet werden kann und ob der erste Wunsch dieser Teilgruppe schon vergeben wurde. Danach wird durch alle zweiten Wünsche iteriert und nach passenden Wünschen gesucht, dies wird danach mit den dritten Wünschen wiederholt.

Hierbei nutze ich zwei Klassen: `Player` und `Wish`, sowie die Enumeration `WishEnum`. Wie die Namen schon verraten können, stellt ein Objekt der Klasse `Player` einen Teilnehmer, ein `Wish`-Objekt einen Wunsch dar und ein Objekt der Enumeration `WishEnum` kann bloß die Werte `FIRST`, `SECOND` und `THIRD` haben.

Player

`std::array<int, 3> wishes`	Speichert die drei Wünsche des Teilnehmers
`int present`	Speichert das zugewiesene Geschenk
`bool assigned`	Speichert, ob versucht wurde, Geschenk zu geben

Hinzu kommen dann noch Konstruktoren, Set- und Get-Methoden, sowie Operatoren-Überladungen zum Sortieren in `std::set`.

Wish

`int wish`	Speichert die Nummer dieses Geschenks
`int wishAmount`	Speichert, wie oft sich das Geschenk gewünscht wird
`std::set<Player*> plys`	Speichert Pointer zu Spielern, die sich dieses Geschenk wünschen
`bool has(Player* ply)`	Überprüft, ob Spieler ply sich dieses Geschenk wünscht und liefert dementsprechend zurück

`bool isPosition(WishEnum pos)`	Überprüft ob sich das Geschenk an der Position (1., 2., 3. Wunsch) von pos gewünscht wird und liefert dementsprechend zurück
`player withPosition(WishEnum pos)`	Überprüft ob sich das Geschenk an der Position (1., 2., 3. Wunsch) von pos gewünscht wird und liefert einen dafür passenden Spieler zurück

Hinzu kommen noch Set- und Get-Methoden für alle Attribute, sowie ein Konstruktor

Ausführung

Folgendes gilt nur für das Betriebssystem Mac OS X (10.15.7)

- Das aktuelle Arbeitsverzeichnis (*pwd*) lautet: */Users/BwInf*
- Die Ausführbare Datei befindet sich im Verzeichnis: */Users/BwInf/2020/Runde1/TheodorTeslia/Aufgabe5/bin*
- In Terminal eingeben: *cd 2020/Runde1/TheodorTeslia/Aufgabe5/*
- Dann: *./Aufgabe5*
  - Durchläuft das Programm mit der Standard-Eingabedatei (wichteln1.txt)
  - Zum testen mit anderen Dateien: *./Aufgabe5 <TextDatei.txt>*
    - *<TextDatei.txt>* muss sich im Verzeichnis */Users/BwInf/2020/Runde1/TheodorTeslia/Aufgabe5/examples* befinden
    - Das Ergebnis wird dann in der Datei */Users/BwInf/2020/Runde1/TheodorTeslia/Aufgabe5/out/<TextDatei.txt>* gespeichert

Beispiele

```
bin % ./Aufgabe5 wichteln1.txt
```

```
Der 1. Teilnehmer bekommt das Geschenk 2 (Dies entspricht dem 1. Wunsch)
Der 2. Teilnehmer bekommt das Geschenk 5 (Dies entspricht keinem Wunsch)
Der 3. Teilnehmer bekommt das Geschenk 1 (Dies entspricht dem 3. Wunsch)
Der 4. Teilnehmer bekommt das Geschenk 3 (Dies entspricht dem 1. Wunsch)
Der 5. Teilnehmer bekommt das Geschenk 8 (Dies entspricht keinem Wunsch)
Der 6. Teilnehmer bekommt das Geschenk 4 (Dies entspricht dem 1. Wunsch)
Der 7. Teilnehmer bekommt das Geschenk 7 (Dies entspricht dem 1. Wunsch)
Der 8. Teilnehmer bekommt das Geschenk 10 (Dies entspricht dem 1. Wunsch)
Der 9. Teilnehmer bekommt das Geschenk 9 (Dies entspricht dem 1. Wunsch)
Der 10. Teilnehmer bekommt das Geschenk 6 (Dies entspricht dem 3. Wunsch)
```

```
bin % ./Aufgabe5 wichteln2.txt
```

```
Der 1. Teilnehmer bekommt das Geschenk 4 (Dies entspricht dem 1. Wunsch)
Der 2. Teilnehmer bekommt das Geschenk 5 (Dies entspricht dem 1. Wunsch)
Der 3. Teilnehmer bekommt das Geschenk 6 (Dies entspricht dem 1. Wunsch)
Der 4. Teilnehmer bekommt das Geschenk 1 (Dies entspricht keinem Wunsch)
Der 5. Teilnehmer bekommt das Geschenk 2 (Dies entspricht keinem Wunsch)
Der 6. Teilnehmer bekommt das Geschenk 3 (Dies entspricht keinem Wunsch)
Der 7. Teilnehmer bekommt das Geschenk 7 (Dies entspricht keinem Wunsch)
Der 8. Teilnehmer bekommt das Geschenk 8 (Dies entspricht keinem Wunsch)
Der 9. Teilnehmer bekommt das Geschenk 9 (Dies entspricht keinem Wunsch)
Der 10. Teilnehmer bekommt das Geschenk 10 (Dies entspricht keinem Wunsch)
```

```
bin % ./Aufgabe5 wichteln3.txt
```

```
Der 1. Teilnehmer bekommt das Geschenk 2 (Dies entspricht dem 1. Wunsch)
Der 2. Teilnehmer bekommt das Geschenk 20 (Dies entspricht dem 1. Wunsch)
Der 3. Teilnehmer bekommt das Geschenk 29 (Dies entspricht dem 1. Wunsch)
Der 4. Teilnehmer bekommt das Geschenk 8 (Dies entspricht dem 1. Wunsch)
Der 5. Teilnehmer bekommt das Geschenk 1 (Dies entspricht keinem Wunsch)
Der 6. Teilnehmer bekommt das Geschenk 3 (Dies entspricht dem 1. Wunsch)
Der 7. Teilnehmer bekommt das Geschenk 5 (Dies entspricht keinem Wunsch)
Der 8. Teilnehmer bekommt das Geschenk 12 (Dies entspricht dem 1. Wunsch)
Der 9. Teilnehmer bekommt das Geschenk 4 (Dies entspricht dem 1. Wunsch)
Der 10. Teilnehmer bekommt das Geschenk 28 (Dies entspricht dem 1. Wunsch)
Der 11. Teilnehmer bekommt das Geschenk 6 (Dies entspricht dem 1. Wunsch)
Der 12. Teilnehmer bekommt das Geschenk 9 (Dies entspricht dem 1. Wunsch)
Der 13. Teilnehmer bekommt das Geschenk 14 (Dies entspricht dem 2. Wunsch)
Der 14. Teilnehmer bekommt das Geschenk 7 (Dies entspricht keinem Wunsch)
Der 15. Teilnehmer bekommt das Geschenk 26 (Dies entspricht dem 1. Wunsch)
Der 16. Teilnehmer bekommt das Geschenk 30 (Dies entspricht dem 1. Wunsch)
Der 17. Teilnehmer bekommt das Geschenk 11 (Dies entspricht keinem Wunsch)
Der 18. Teilnehmer bekommt das Geschenk 13 (Dies entspricht keinem Wunsch)
Der 19. Teilnehmer bekommt das Geschenk 16 (Dies entspricht dem 1. Wunsch)
Der 20. Teilnehmer bekommt das Geschenk 10 (Dies entspricht dem 1. Wunsch)
Der 21. Teilnehmer bekommt das Geschenk 15 (Dies entspricht keinem Wunsch)
Der 22. Teilnehmer bekommt das Geschenk 17 (Dies entspricht keinem Wunsch)
Der 23. Teilnehmer bekommt das Geschenk 27 (Dies entspricht dem 1. Wunsch)
Der 24. Teilnehmer bekommt das Geschenk 18 (Dies entspricht keinem Wunsch)
Der 25. Teilnehmer bekommt das Geschenk 19 (Dies entspricht dem 2. Wunsch)
Der 26. Teilnehmer bekommt das Geschenk 21 (Dies entspricht keinem Wunsch)
Der 27. Teilnehmer bekommt das Geschenk 22 (Dies entspricht keinem Wunsch)
Der 28. Teilnehmer bekommt das Geschenk 23 (Dies entspricht keinem Wunsch)
Der 29. Teilnehmer bekommt das Geschenk 24 (Dies entspricht keinem Wunsch)
```

```
Der 30. Teilnehmer bekommt das Geschenk 25 (Dies entspricht keinem Wunsch)
```

```
bin % ./Aufgabe5 wichteln4.txt
```

```
Der 1. Teilnehmer bekommt das Geschenk 28 (Dies entspricht dem 1. Wunsch)
Der 2. Teilnehmer bekommt das Geschenk 21 (Dies entspricht dem 1. Wunsch)
Der 3. Teilnehmer bekommt das Geschenk 14 (Dies entspricht dem 1. Wunsch)
Der 4. Teilnehmer bekommt das Geschenk 4 (Dies entspricht keinem Wunsch)
Der 5. Teilnehmer bekommt das Geschenk 5 (Dies entspricht keinem Wunsch)
Der 6. Teilnehmer bekommt das Geschenk 7 (Dies entspricht keinem Wunsch)
Der 7. Teilnehmer bekommt das Geschenk 3 (Dies entspricht dem 2. Wunsch)
Der 8. Teilnehmer bekommt das Geschenk 9 (Dies entspricht keinem Wunsch)
Der 9. Teilnehmer bekommt das Geschenk 10 (Dies entspricht keinem Wunsch)
Der 10. Teilnehmer bekommt das Geschenk 19 (Dies entspricht dem 1. Wunsch)
Der 11. Teilnehmer bekommt das Geschenk 11 (Dies entspricht keinem Wunsch)
Der 12. Teilnehmer bekommt das Geschenk 12 (Dies entspricht dem 1. Wunsch)
Der 13. Teilnehmer bekommt das Geschenk 16 (Dies entspricht keinem Wunsch)
Der 14. Teilnehmer bekommt das Geschenk 20 (Dies entspricht keinem Wunsch)
Der 15. Teilnehmer bekommt das Geschenk 2 (Dies entspricht dem 1. Wunsch)
Der 16. Teilnehmer bekommt das Geschenk 23 (Dies entspricht keinem Wunsch)
Der 17. Teilnehmer bekommt das Geschenk 1 (Dies entspricht dem 1. Wunsch)
Der 18. Teilnehmer bekommt das Geschenk 27 (Dies entspricht dem 1. Wunsch)
Der 19. Teilnehmer bekommt das Geschenk 17 (Dies entspricht dem 2. Wunsch)
Der 20. Teilnehmer bekommt das Geschenk 13 (Dies entspricht dem 1. Wunsch)
Der 21. Teilnehmer bekommt das Geschenk 22 (Dies entspricht dem 2. Wunsch)
Der 22. Teilnehmer bekommt das Geschenk 25 (Dies entspricht keinem Wunsch)
Der 23. Teilnehmer bekommt das Geschenk 15 (Dies entspricht dem 1. Wunsch)
Der 24. Teilnehmer bekommt das Geschenk 26 (Dies entspricht keinem Wunsch)
Der 25. Teilnehmer bekommt das Geschenk 30 (Dies entspricht dem 1. Wunsch)
Der 26. Teilnehmer bekommt das Geschenk 24 (Dies entspricht dem 1. Wunsch)
Der 27. Teilnehmer bekommt das Geschenk 18 (Dies entspricht dem 1. Wunsch)
Der 28. Teilnehmer bekommt das Geschenk 8 (Dies entspricht dem 1. Wunsch)
Der 29. Teilnehmer bekommt das Geschenk 6 (Dies entspricht dem 2. Wunsch)
Der 30. Teilnehmer bekommt das Geschenk 29 (Dies entspricht dem 1. Wunsch)
```

```
bin % ./Aufgabe5 wichteln5.txt
```

```
Der 1. Teilnehmer bekommt das Geschenk 5 (Dies entspricht keinem Wunsch)
Der 2. Teilnehmer bekommt das Geschenk 6 (Dies entspricht dem 1. Wunsch)
Der 3. Teilnehmer bekommt das Geschenk 7 (Dies entspricht dem 1. Wunsch)
Der 4. Teilnehmer bekommt das Geschenk 8 (Dies entspricht keinem Wunsch)
Der 5. Teilnehmer bekommt das Geschenk 27 (Dies entspricht dem 3. Wunsch)
Der 6. Teilnehmer bekommt das Geschenk 2 (Dies entspricht dem 1. Wunsch)
```

```
Der 7. Teilnehmer bekommt das Geschenk 4 (Dies entspricht dem 1. Wunsch)
Der 8. Teilnehmer bekommt das Geschenk 18 (Dies entspricht dem 1. Wunsch)
Der 9. Teilnehmer bekommt das Geschenk 9 (Dies entspricht keinem Wunsch)
Der 10. Teilnehmer bekommt das Geschenk 16 (Dies entspricht keinem Wunsch)
Der 11. Teilnehmer bekommt das Geschenk 14 (Dies entspricht dem 1. Wunsch)
Der 12. Teilnehmer bekommt das Geschenk 13 (Dies entspricht dem 1. Wunsch)
Der 13. Teilnehmer bekommt das Geschenk 19 (Dies entspricht dem 3. Wunsch)
Der 14. Teilnehmer bekommt das Geschenk 15 (Dies entspricht dem 1. Wunsch)
Der 15. Teilnehmer bekommt das Geschenk 10 (Dies entspricht dem 1. Wunsch)
Der 16. Teilnehmer bekommt das Geschenk 20 (Dies entspricht keinem Wunsch)
Der 17. Teilnehmer bekommt das Geschenk 26 (Dies entspricht dem 3. Wunsch)
Der 18. Teilnehmer bekommt das Geschenk 23 (Dies entspricht dem 3. Wunsch)
Der 19. Teilnehmer bekommt das Geschenk 21 (Dies entspricht keinem Wunsch)
Der 20. Teilnehmer bekommt das Geschenk 3 (Dies entspricht dem 3. Wunsch)
Der 21. Teilnehmer bekommt das Geschenk 1 (Dies entspricht dem 1. Wunsch)
Der 22. Teilnehmer bekommt das Geschenk 22 (Dies entspricht keinem Wunsch)
Der 23. Teilnehmer bekommt das Geschenk 24 (Dies entspricht keinem Wunsch)
Der 24. Teilnehmer bekommt das Geschenk 11 (Dies entspricht dem 1. Wunsch)
Der 25. Teilnehmer bekommt das Geschenk 28 (Dies entspricht keinem Wunsch)
Der 26. Teilnehmer bekommt das Geschenk 29 (Dies entspricht keinem Wunsch)
Der 27. Teilnehmer bekommt das Geschenk 30 (Dies entspricht dem 3. Wunsch)
Der 28. Teilnehmer bekommt das Geschenk 12 (Dies entspricht dem 1. Wunsch)
Der 29. Teilnehmer bekommt das Geschenk 17 (Dies entspricht dem 1. Wunsch)
Der 30. Teilnehmer bekommt das Geschenk 25 (Dies entspricht dem 3. Wunsch)
```

```
bin % ./Aufgabe5 wichteln6.txt
```

Siehe die Ausgabedatei (90 Zeilen wären doch etwas zu viel)

```
bin % ./Aufgabe5 wichteln7.txt
```

Siehe die Ausgabedatei (1000 Zeilen würden den Rahmen hier schon sehr sprengen)

Quellcode

```
        for (auto& i : plyVec) {  // Fill first wishes
                wishes.at(FIRST).insert(i.getWish(FIRST));
                alreadyUsed.at(i.getWish(FIRST)) += 1;
        }
        for (auto& i : plyVec) {  // Fill second wishes
                if ( wishes.at(FIRST).find(i.getWish(SECOND)) == wishes.at(FIRST).end() &&  // Number doesn't
occur in higher List
                                alreadyUsed.at(i.getWish(FIRST)) != 1) {
                        wishes.at(SECOND).insert(i.getWish(SECOND));  // ^ False if not clearly assigned
```

```
                        alreadyUsed.at(i.getWish(SECOND)) += 1;
                }
        }
        for (auto& i : plyVec) {  // Fill third wishes
                if (wishes.at(FIRST).find(i.getWish(THIRD)) == wishes.at(FIRST).end() &&
                        wishes.at(SECOND).find(i.getWish(THIRD)) == wishes.at(SECOND).end() &&
                        alreadyUsed.at(i.getWish(FIRST)) != 1) {
                wishes.at(THIRD).insert(i.getWish(THIRD));
                alreadyUsed.at(i.getWish(THIRD)) += 1;
                }
        }



Player* findPlayerWithLowestWishes(std::vector< std::pair<Player*, int> >& vec, std::vector<Wish>& list,
std::map<int, int>& used, bool& specialSecond, std::set<int>& notToBeUsedAgain) {
        std::pair<Player*, int> lowestPair;  // Set up the starting player
        int counter = 0;  // Starting player is not allowed to have already a fulfilled present
        do {  // Else this leads to error
                lowestPair.first = vec.at(counter).first;
                lowestPair.second = vec.at(counter).second;
                counter++;
        } while (lowestPair.first->wasAssigned() && counter <= vec.size());


        Player* lowestPlayer = lowestPair.first;
        int lowestAmount = lowestPair.second;
        for (auto& i : vec) {  // Find person with least wishes fulfillable and who didn't get a present yet
                if (i.second < lowestAmount && i.second != 0 && i.first->getPresent() == -1 && !lowestPlayer-
>wasAssigned()) {
                        lowestAmount = i.second;
                        lowestPlayer = i.first;
                }
        }
        if (used.at(lowestPlayer->getWish(SECOND)) == 1 && notToBeUsedAgain.find(lowestPlayer->getWish(SECOND))
== notToBeUsedAgain.end() && !lowestPlayer->wasAssigned() && find(list, lowestPlayer->getWish(SECOND))) {  //
TODO: Do something with the special second
                specialSecond = true;
        }
        return lowestPlayer;
}



int whichWishToBeUsed(Player*& ply, std::vector<Wish>& list, std::set<int>& used, bool state, bool specialSecond,
std::vector< std::pair<Player*, int> >& plyList) {  // TODO: Wooork!
```

```
        if (specialSecond && state) {
                return ply->getWish(SECOND);
        }
        if (!state) {
                return ply->getWish(FIRST);
        }

        for (auto& i : list) {  // Use a fitting second wish
                if (used.find(i.getWish()) == used.end() && i.getWish() == ply->getWish(SECOND)) {
                        return i.getWish();
                }
        }
        for (auto& i : list) {  // Use a fitting third wish
                if (used.find(i.getWish()) == used.end() && i.getWish() == ply->getWish(THIRD)) {
                        return i.getWish();
                }
        }

        return -1;
}
```