Aufgabe 1: Rosinen picken

**Ansatz**

Ein Unternehmen besteht aus folgenden Informationen:

- Einer ID

- Einem Wert

- Unternehmen, auf die verwiesen wird

- Unternehmen, die auf das Unternehmen verweisen

Im Folgenden ist ein Kindunternehmen eines anderen Unternehmens ein Unternehmen, das nur mit dem anderen Unternehmen gekauft werden kann.

Ein Elternunternehmen ist ein Unternehmen, das, sollte es gekauft werden, den Kauf des Unternehmens ebenfalls nötig macht.

Bsp. (A → B → C)

A ist Elternunternehmen von B.

B ist Kindunternehmen von A.

Das Programm erstellt eine Liste von Unternehmen aus der Eingabedatei, die jeweils Verweise auf ihre Kind- und Elternunternehmen haben. Außerdem hat jedes Unternehmen eine Angabe, ob es bereits geprüft wurde und ob es zum Kauf markiert wurde.

Es wird nun bei einer elternloser Firma angesetzt (sollte keine solche existieren, wird beim ersten Unternehmen der Liste angesetzt) und so lange das erste Kindunternehmen ausgewählt, das noch nicht überprüft wurde, bis ein Unternehmen keine nicht bereits geprüfte Kindunternehmen hat.

Es wird als überprüft markiert.

Für das Unternehmen und alle Nachkommenunternehmen davon, die nicht bereits zum Kauf gewählt wurden, wird dann der Gesamtwert berechnet und, sollte er positiv sein, werden es und alle Nachkommenunternehmen zum Kauf markiert.

Sollte dabei ein Unternehmen mit negativem Wert zum Kauf markiert werden, werden alle Vorfahrenunternehmen davon, die nicht bereits zum Kauf markiert sind, zur erneuten Überprüfung gekennzeichnet, da sie dadurch eventuell einen positiven Gesamtwert erhalten.

Dann wird eine Ebene im Pfad nach oben gegangen und erneut ein kinderloses Unternehmen gesucht. Sollte der Pfad keine übergeordneten Unternehmen haben, wird bei einem anderen elternlosen Unternehmen (oder einem beliebigen) angesetzt. Dies wird so lange wiederholt, bis alle Unternehmen geprüft wurden.

Sonderfall: Kreisbezug

Es kann vorkommen, dass Unternehmen im Kreis aufeinander verweisen. In diesem Fall wird der Kreisbezug aus dem Pfad extrahiert und es wird geprüft, ob sich die Übernahme aller Unternehmen im Kreis lohnt.

**Umsetzung**

Das Programm ist eine Windows Forms Anwendung, die nach Klicken auf Datei → Laden eine Dateieingabe ermöglicht und nach Ende der Berechnungen automatisch den Standardeditor für \*.txt Dateien öffnet. Die Ausgabedateien finden sich im Programmverzeichnis unter {Eingabedatei}\_result.txt

private void LoadFile(string path)

{

using(var reader = new StreamReader(path))

{

//Zurücksetzen

Companies = new List<Company>();

Path = new Stack<int>();

//Erste Zeilen überspringen

reader.ReadLine();

reader.ReadLine();

reader.ReadLine();

while (reader.Peek() != '#')

{

string line = reader.ReadLine();

var linsSplit = line.Split(' ');

Companies.Add(new Company(

Convert.ToInt32(linsSplit[0]),

float.Parse(linsSplit[1], CultureInfo.InvariantCulture.NumberFormat)));

}

reader.ReadLine(); //Zeile überspringen

while (reader.Peek() != -1)

{

string line = reader.ReadLine();

var lineSplit = line.Split(' ');

int[] ids = { Convert.ToInt32(lineSplit[0]), Convert.ToInt32(lineSplit[1]) };

Companies[ids[0]].Children.Add(ids[1]); // Unternehmen als Kind registrieren

Companies[ids[1]].Parents.Add(ids[0]); // Unternehmen als Elternteil registrieren

}

}

}

public void Start()

{

Path.Clear();

Path.Push(Companies.Any(x => x.Parents.Count == 0)

? (Companies.First(x => x.Parents.Count == 0)).Id : 0);

//An Unternehmen ohne Elternunternehmen ansetzen, sonst erstes Unternehmen der Liste wählen

Companies = Companies.OrderBy(x => x.Id).ToList();

//Liste sortieren => Id entspricht Position in Liste

while (!Companies.All(x => x.IsChecked))

{

FallToChildless();

if (ValueOfList(GetChildCompanies(Path.Peek())) > 0)

PickList(GetChildCompanies(Path.Pop()));

else

{

Companies[Path.Pop()].IsChecked = true;

}

}

//Ausgabe

using (var writer = File.CreateText(

fileInput.FileName.Insert(fileInput.FileName.Length - 4, "\_result")))

{

var result = Companies.Where(x => x.IsPicked).ToList();

writer.WriteLine($"# Ergebnis für Datei {fileInput.FileName}");

writer.WriteLine("# Anzahl der Unternehmen");

writer.WriteLine(result.Count());

writer.WriteLine("# Gesamtwert der Unternehmen");

writer.WriteLine(result.Sum(x => x.Value));

writer.WriteLine("# Nummern der Unternehmen");

foreach (var c in result.OrderBy(x => x.Id))

writer.WriteLine(c.Id); //Geordnete Nummernausgabe

}

Process.Start(fileInput.FileName.Insert(fileInput.FileName.Length - 4, "\_result"));

}

public void FallToChildless()

{

if (Path.Count == 0)

{

//Neu ansetzen

Path.Push(Companies.Any(x => !x.Parents.Any(xp => Companies[xp].IsChecked) && !x.IsChecked) //An elternlosem Element, sonst an beliebigem ungeprüftem Element

? Companies.First(x => !x.Parents.Any(xp => Companies[xp].IsChecked) && !x.IsChecked).Id

: Companies.First(x => !x.IsChecked).Id);

}

while (Companies[Path.Peek()].Children.Count > 0)

{

if (Companies[Path.Peek()].Children.All(x => Companies[x].IsChecked))

//Wenn alle Kindelemente gerpüft wurden, wurde das Ziel erreicht.

break;

var id = Companies[Path.Peek()].Children.First(x => !Companies[x].IsChecked);

if (Path.Contains(id)) //Kreisbezug => Auswahl prüfen

{

var list = new List<int>();

while (Path.Peek() != id)

list.Add(Path.Pop());

if (ValueOfList(list) > 0)

PickList(list);

else

foreach (var l in list)

Companies[l].IsChecked = true;

}

else

Path.Push(id);

}

}

public List<int> GetChildCompanies(int id)

{

var linkedCompanies = new List<int>();

var path = new Stack<int>();

path.Push(id);

while (!linkedCompanies.Contains(id) || path.Count > 1)

//Solange noch nicht alle Kindelelemente besucht wurden

{

if (Companies[path.Peek()].Children.Any(x => !linkedCompanies.Contains(x) && ! Companies[x].IsPicked))

{

var next = Companies[path.Peek()].Children.First(x => !linkedCompanies.Contains(x) && ! Companies[x].IsPicked);

if (path.Contains(next)) //Kreis geschlossen

linkedCompanies.Add(next); //Auflösen

else

path.Push(next); //Ebene nach unten

}

else

{

linkedCompanies.Add(path.Pop());

}

}

return linkedCompanies;

}

public List<int> GetParentCompanies(int id)

{

var linkedCompanies = new List<int>();

var path = new Stack<int>();

path.Push(id);

while (!linkedCompanies.Contains(id) || path.Count > 1)

//Solange noch nicht alle Kindelelemente besucht wurden

{

if (Companies[path.Peek()].Parents.Any(x => !linkedCompanies.Contains(x) && ! Companies[x].IsPicked))

{

var next = Companies[path.Peek()].Parents.First(x => !linkedCompanies.Contains(x) && ! Companies[x].IsPicked);

if (path.Contains(next)) //Kreis geschlossen

linkedCompanies.Add(next); //Auflösen

else

path.Push(next); //Ebene nach "unten"

}

else

{

linkedCompanies.Add(path.Pop());

}

}

return linkedCompanies;

}

/// <summary>

/// Gibt den Wert aller noch nicht gewählter Unternehmen der Liste zurück

/// </summary>

/// <param name="ids">Liste an Unternehmen-IDs</param>

/// <returns></returns>

public float ValueOfList(List<int> ids)

{

return ids.Where(c => !Companies[c].IsPicked).Sum(c => Companies[c].Value);

}

/// <summary>

/// Markiert eine Liste von Unternehmen zum Kauf

/// </summary>

/// <param name="ids">Liste an Unternehmen-IDs</param>

public void PickList(List<int> ids)

{

foreach (var c in ids)

{

if (Companies[c].IsPicked)

throw new Exception($"Unternehmen mit Id {Companies[c].Id} bereits gewählt!");

Companies[c].IsPicked = true;

Companies[c].IsChecked = true;

}

foreach (var c in ids)

{

if (!(Companies[c].Value < 0))

continue;

foreach (var reactivate in GetParentCompanies(c)

//Wenn der Wert eines Unternehmens negativ ist, die nicht gewählten Elternelemente neu prüfen lassen

.Where(x => Companies[x].IsChecked && !Companies[x].IsPicked))

{

Companies[reactivate].IsChecked = false;

}

}

}

**Beispiele**

Die Ergebnisse zu den Beispielen im vorgegebenen Ausgabeformat finden sich im Order „Aufgabe 1-Ergebnisse“.

**Laufzeit**

Die Laufzeit des Programms hängt von verschiedenen Faktoren ab. Neben der Anzahl der Unternahmen spielt auch der Aufbau der Verbindungen eine große Rolle. Das entsteht durch die erneute Überprüfung, wenn ein Element gewählt wird. Dennoch ist die Laufzeit nie ein Problem, da eine Unternehmenszahl über 1000 unrealistisch ist und der Aufbau der Verbindungen sehr ungünstig sein muss, um die Laufzeit merkbar zu beeinflussen.