Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

186.815 Algorithmen und Datenstrukturen 2 VU 3.0

Sommersemester 2012

Programmieraufgabe

abzugeben bis: Freitag, 8. Juni 2012, 15:00 Uhr

Organisatorisches

Im Rahmen der Lehrveranstaltung **Algorithmen und Datenstrukturen 2** gilt es, eine Programmieraufgabe selbständig zu lösen, die das Verständnis des im Vorlesungsteil vorgetragenen Stoffes vertiefen soll. Als Programmiersprache wird Java 6 verwendet.

Geben Sie bitte Ihr fertiges, gut getestetes und selbst geschriebenes Programm bis spätestens **Freitag, 8. Juni 2012, 15:00 Uhr** über das Abgabesystem in TUWEL ab. Der von Ihnen abgegebene Code wird vom System automatisch getestet, und Sie erhalten eine entsprechende Rückmeldung im Abgabesystem.

Um eine positive Note zu erhalten, müssen Sie bei der Programmieraufgabe **mindestens einen Punkt erreichen** und Ihren Termin zum Abgabegespräch einhalten.

> Gruppenabgaben bzw. mehrfache Abgaben desselben Programms unter verschiedenen Namen werden nicht akzeptiert. Wenn Sie das abgegebene Programm nicht selbst programmiert haben, erhalten Sie ein negatives Zeugnis. Auch der eigentliche Entwickler kann nur mehr maximal einen Punkt auf dieses Programmierbeispiel erhalten.

Abgabefrist

Sie haben bis Freitag, 8. Juni 2012, 15:00 Uhr die Möglichkeit ein Programm abzugeben, **das alle Testinstanzen korrekt abarbeitet**. Danach werden keine weiteren Abgaben akzeptiert, d.h., sollten Sie diese Frist versäumen beziehungsweise Ihr Programm nicht alle Testinstanzen korrekt abarbeiten, bekommen Sie keine Punkte auf die Programmieraufgabe.

Abgabegespräche

Am Dienstag, dem 12. Juni 2012, Donnerstag, dem 14. Juni 2012, und Freitag, dem 15. Juni 2012, finden für alle LVA-Teilnehmer Abgabegespräche statt. Dazu vereinbaren Sie in TUWEL einen individuellen Gesprächstermin, bei dem Sie sich mit einer/m unserer TutorInnen

im Informatiklabor treffen und den von Ihnen eingereichten Programmcode erklären können müssen. Sie können sich zu diesem Abgabegespräch von 8. Juni 2012 bis 11. Juni 2012, 15:00 Uhr in TUWEL bei einer/m TutorIn anmelden. Sollten in diesem Zeitraum keine Termine in TUWEL eingetragen oder bereits alle Termine vergeben sein, dann kontaktieren Sie bitte die Hotline `algodat2-ss12@ads.tuwien.ac.at`.
Melden Sie sich nur an, wenn Sie positiv abgegeben haben!

Falls Sie Systemroutinen in Ihrem Programm verwenden (z.B. das Auffinden eines Minimums), so sollten Sie auch über die Funktionsweise dieser Methoden **genau** Bescheid wissen. Je nach Funktionstüchtigkeit Ihres Programms, Innovation und Effizienz Ihrer Implementierung sowie der Qualität des Abgabegesprächs können Sie bis zu 20 Punkte für diese Programmieraufgabe erhalten. Voraussetzung für eine positive Absolvierung des Abgabegesprächs ist jedenfalls das Verständnis des zugrunde liegenden Stoffes.

Betreuung im Informatiklabor

Um auftretende Verständnisfragen beziehungsweise Unklarheiten bei der Angabe klären zu können, stehen Ihnen am Montag, dem 04. Juni 2012, von 12:30 bis 13:30 und am Mittwoch, dem 06. Juni 2012, von 12:30 bis 13:30 unsere Tutoren im Informatiklabor für persönliche Gespräche zur Verfügung. Bedenken Sie allerdings, dass dies nur eine Hilfestellung ist und eine selbständige Auseinandersetzung mit der Problemstellung nicht ersetzen soll und kann – die TutorInnen sind nicht als „menschliche Debugger“ anzusehen.

Aufgabenstellung

Eine Flotte von Booten wird von einem Energiekonzern ausgeschickt um eine Küstenregion genau zu vermessen. Ziel ist es mögliche Standorte für die Installation von Windkraftanlagen zu finden. Nachdem die Mission abgeschlossen ist, liegt eine Karte dieser Küstenregion vor, auf der n potentielle Standorte markiert sind. Der Energiekonzern möchte das Gebiet mit k $(0 < k < n)$ Windrädern erschließen, wobei die Standorte so ausgewählt werden sollen, dass die Kosten für die zu verlegenden Stromleitungen minimiert werden.

Der Energiekonzern kann das Problem formal folgendermaßen definieren:
Gegeben ist eine Konstante k und ein gewichteter, ungerichteter Graph $G = (V, E)$ mit Knotenmenge V, Kantenmenge E und Kantenkosten $c_{ij},\ \forall (i, j) \in E$, die den Verlegekosten von i nach j entsprechen, $i, j \in V$. Sei $n = |V|$, dann gilt $0 < k < n$.
Gesucht ist ein minimaler Baum, der genau k Knoten in G verbindet. Dieser wird auch k-minimaler Spannbaum oder k-MST genannt.

An dieser Stelle weiß der Energiekonzern aber nicht mehr weiter und bittet Sie um Hilfe. Aus Algorithmen und Datenstrukturen 1 wissen Sie, dass für den Fall $k = n$ das Problem in polynomieller Zeit exakt gelöst werden kann. Für $0 < k < n$ ist das Problem allerdings $\mathcal{NP}$-schwer. Da Sie vor Kurzem in Algorithmen und Datenstrukturen 2 das Branch-and-Bound Verfahren kennengelernt haben, entschließen Sie sich das Problem auf diese Weise exakt zu lösen.

Hierfür entwickeln und implementieren Sie jeweils einen Algorithmus zur Berechnung einer heuristischen Lösung und damit verbunden einer oberen Schranke, sowie einer unteren Schranke. Diese Algorithmen verwenden Sie in einem Branch-and-Bound Verfahren.

Konkret sieht Ihre Aufgabenstellung wie folgt aus:

1. Überlegen Sie, wie Sie das Branching ausführen, d.h., wie Sie ein (Unter-)Problem in weitere Unterprobleme zerteilen, und mithilfe welcher Datenstrukturen Sie diese (Unter-)Probleme speichern.

2. Entwickeln Sie darauf aufbauend einen heuristischen Algorithmus, um möglicherweise für ein (Unter-)Problem eine gültige Lösung zu erhalten, deren Wert eine globale obere Schranke darstellt.

3. Entwickeln Sie weiters eine Dualheuristik, die für jedes (Unter-)Problem eine untere Schranke liefert.

4. Für die grundsätzliche Funktionsweise des Branch-and-Bound ist es egal, welches offene Unterproblem aus der Problemliste ausgewählt und als nächstes abgearbeitet wird. In der Praxis spielt diese Auswahlstrategie jedoch in Bezug auf die Laufzeit eine große Rolle. Wir schlagen hier vor, eine einfache Depth-First-Strategie zu implementieren, bei der nach einem Branching immer eines der neu erzeugten Unterprobleme als unmittelbar nächstes abgearbeitet wird. In dieser Weise kann das Branch-and-Bound direkt als rekursiver Algorithmus ohne zusätzliche Datenstrukturen zur expliziten Speicherung der Problemliste implementiert werden.

5. Implementieren Sie nun das vollständige Branch-and-Bound, das auf den oben genannten Punkten aufbaut.

Hinweis zur Laufzeit

Pro Instanz stehen Ihrem Programm am Abgabeserver maximal 30 Sekunden CPU-Zeit zur Verfügung. Findet Ihr Algorithmus in dieser Zeit keine optimale Lösung, dann wird die beste bisher gefundene, gültige Lösung zur Bewertung herangezogen.

Codegerüst

Der Algorithmus ist in Java 6 zu implementieren. Um Ihnen einerseits das Lösen der Aufgabenstellung zu erleichtern und andererseits automatisches Testen mit entsprechender Rückmeldung an Sie zu ermöglichen, stellen wir Ihnen ein Codegerüst zur Verfügung[1].

Sie können das Framework wie folgt kompilieren:

```
$ javac ads1ss12/pa/*.java
```

Das Codegerüst besteht aus mehreren Klassen, wobei Sie Ihren Code in die Datei `KMST.java` einfügen müssen.

Weitere Details, die das Codegerüst betreffen, entnehmen Sie bitte der LVA-Webseite[2]. Dort befindet sich auch der Link zur Dokumentation (Javadoc) des Codegerüstes.

Hinweise

`Main.printDebug(String msg)` kann verwendet werden, um Debuginformationen auszugeben. Die Ausgabe erfolgt nur, wenn beim Aufrufen das Debugflag (`-d`) gesetzt wurde. Sie müssen diese Ausgaben vor der Abgabe **nicht** entfernen, da das Testsystem beim Kontrollieren Ihrer Implementierung dieses Flag nicht setzt.

Sie müssen die Methode `boolean setSolution(int newUpperBound, Set<Edge> newSoluton)` der Klasse `AbstractKMST` verwenden, um dem Framework eine neue (beste) Lösung bekannt zu geben. Die Lösung wird nur übernommen, wenn ihr Wert eine verbesserte (d.h. neue niedrigste) obere Schanke darstellt. Die Methode liefert `true` zurück, wenn die Lösung übernommen wurde. **Achtung:** der k-MST wird von der Methode nicht überprüft. Sie müssen sicher stellen, dass es sich um eine korrekte Lösung handelt.

Das Framework nutzt Methoden, die das Ausführen von *sicherheitsbedenklichem* Code auf dem Abgabesystem verhindern soll. Werden trotzdem solche Programmteile abgegeben oder sollte versucht werden, das Sicherheitssystem zu umgehen, wird das als Betrugsversuch gewertet. Die minimale Konsequenz dafür ist ein negatives Zeugnis auf diese Lehrveranstaltung.

[1] `http://www.ads.tuwien.ac.at/teaching/lva/186815.html`
[2] `http://www.ads.tuwien.ac.at/teaching/lva/186815.html`

Rückgabe des Frameworks

Nach dem Aufruf Ihres Branch-and-Bound Verfahrens wird der zurückgelieferte Spannbaum auf dessen Korrektheit geprüft und das Gesamtgewicht berechnet. Für eine positive Abgabe muss Ihr Verfahren für jede Instanz eine Lösung mit einem Gesamtgewicht liefern, das einen vorgegebenen Schwellwert nicht überschreitet. Wenn ihre Lösung in diesem Sinne ausreichend gut ist, wird vom Framework die Meldung „Ihr Wert ist unter dem Schwellwert“ sowie das Gesamtgewicht Ihres Spannbaums zurückgegeben:

```
Ihr Wert ist unter dem Schwellwert
543
```

Ist das Ergebnis Ihres Verfahrens nicht gut genug, werden Sie Meldungen wie die folgende sehen:

```
ERR zu schlechte Loesung: Ihr Ergebnis 876 liegt ueber dem Schwellwert (765)
```

Liefert Ihr Verfahren `null` oder ein ungültiges Ergebnis zurück, sehen Sie eine Fehlermeldung wie diese:

```
ERR keine gueltige Loesung!
```

Kommandozeilenoptionen Um Ihnen die Arbeit ein wenig zu erleichtern, gibt es Kommandozeilenoptionen, die das Framework veranlassen, mehr Informationen auszugeben.

Wenn Sie beim Aufrufen von Main `-t` angeben, wird die Laufzeit Ihrer Implementierung gemessen. Diese ist natürlich computerabhängig, kann aber trotzdem beim Finden von ineffizienten Algorithmen helfen.

```
$ java ad1ss12.pa.Main -t tests/input/angabe

tests/input/angabe: Ihr Wert ist unter dem Schwellwert
543, Zeit: 15 ms
```

Um zu verhindern, dass das Framework Ihren Algorithmus nach 30 Sekunden beendet, können Sie beim Aufrufen von Main `-s` angeben. Dies ist vor allem zum Debuggen nützlich, da sonst nach dem Ablaufen der Zeit kein weiteres schrittweises Abarbeiten mehr möglich ist.

Testdaten

Auf der Webseite zur LVA sind auch Testdaten veröffentlicht, die es Ihnen erleichtern sollen, Ihre Implementierung zu testen. Verarbeitet Ihr Programm diese Daten korrekt, heißt das aber nicht zwangsläufig, dass Ihr Programm alle Eingaben korrekt behandelt. Testen Sie daher auch mit zusätzlichen, selbst erstellten Daten. Das Abgabesystem wird Ihr Programm mit den öffentlichen und zusätzlich mit nicht veröffentlichten Daten testen.

Die ersten vier Zeilen der Testdaten enthalten die Anzahl der Knoten, die Anzahl der Kanten, den Wert k und den Schwellwert, der jedenfalls erreicht werden muss (in dieser Reihenfolge). Die weiteren Zeilen enthalten die Kanteninformationen, d.h. folgende durch Leerzeichen getrennte Werte: `kantennummer`, `startknotennummer`, `endknotennummer` und `kantengewicht`.

Abgabe

Die Abgabe Ihrer Implementierung der Klasse `KMST` erfolgt über TUWEL. Bedenken Sie bitte, dass Sie maximal 30 Mal abgeben können und ausschließlich die jeweils letzte Abgabe bewertet wird. Prinzipiell sollte es nicht nötig sein, mehr als eine Abgabe zu tätigen, wenn Sie Ihr Programm entsprechend getestet haben.

Hinweis

Verwenden Sie bei Ihrer Implementierung unter keinen Umständen Sonderzeichen, wie zum Beispiel Umlaute (ä, ö, ü, Ä, Ö, Ü) oder das Zeichen „ß“. Dies kann sonst – bei unterschiedlichen Zeichensätzen am Entwicklungs- und Abgabesystem – zu unvorhersehbaren Problemen und Fehlern führen, was schlussendlich auch als fehlerhafter Abgabeversuch gewertet wird.

Die Überprüfung Ihres abgegebenen Codes erfolgt automatisch, wobei in drei Schritten getestet wird:

Kompilation: Es wird der **Bytecode** erzeugt, der beim anschließenden Testen verwendet wird.

Veröffentlichte Testdaten: Bei diesem Schritt wird Ihr Programm mit den auf der Webseite veröffentlichten Daten getestet.

Unveröffentlichte Testdaten: Abschließend wird Ihr Programm noch mit Ihnen nicht bekannten aber den Spezifikationen entsprechenden Eingabedaten ausgeführt.

Nach Beendigung der Tests, die direkt nach Ihrer Abgabe gestartet werden, erhalten Sie eine Rückmeldung über das Abgabesystem in TUWEL. Da es bei dieser Aufgabe mehrere gültige Lösungen geben kann, werden diese je nach Qualität in Kategorien eingeteilt. Ein grünes Zeichen im Abgabesystem bedeutet, dass Ihre Lösung für die jeweilige Instanz sehr gut ist, ein gelbes Zeichen zeigt Ihnen, dass Ihre Lösung zwar ausreichend ist, um positiv

bewertet zu werden. Falls Ihre Lösung für eine Instanz ungültig ist oder sich nicht unterhalb des vorgegebenen Schwellwerts befindet, sehen Sie ein rotes Zeichen. In diesem Fall müssen Sie Ihren Algorithmus noch verbessern, um eine positive Bewertung zu erhalten.

Aufgrund der großen Hörerzahl kann es zu Verzögerungen beim Verarbeiten Ihrer Abgaben kommen. Geben Sie daher Ihre Lösung nicht erst in den letzten Stunden ab, sondern versuchen Sie, rechtzeitig die Aufgabenstellung zu lösen. Beachten Sie bitte auch, dass wir Ihren Code mit den von Ihren Kollegen abgegebenen Programmen automatisch vergleichen werden, um Plagiate zu erkennen. Geben Sie daher nur selbst implementierte Lösungen ab!

Theoriefragen

Beim Abgabegespräch müssen Sie u.a. Ihre Überlegungen zu folgenden Punkten präsentieren können:

- Warum ist beim Branch-and-Bound für Minimierungsprobleme eine obere Schranke global gültig, eine untere Schranke jedoch nur für das entsprechende Subproblem?
- Wann weiß man beim Branch-and-Bound, dass die beweisbar beste Lösung gefunden wurde?
- Wie führen Sie das Branching aus und wie sieht Ihr Suchbaum aus?

Testumgebung

Unser Testsystem mit Intel Xeon X5650 CPU ruft Ihr Programm mit folgendem Kommandozeilenbefehl auf:

```
java ads1ss12.pa.Main input > output
```

wobei `input` eine Eingabedatei darstellt. Die Ausgabe wird in die Datei `output` gespeichert. Pro Testinstanz darf eine maximale Ausführungszeit von 30 Sekunden nicht überschritten werden, anderenfalls wird die Ausführung abgebrochen.

Bewertung

Abschließend seien nochmals die wesentlichen Punkte zusammengefasst, die in die Bewertung Ihrer Abgabe einfließen und schließlich über die Anzahl der Punkte für diese Programmieraufgabe entscheiden:

- Korrektheit des Algorithmus, d.h. alle Instanzen erfolgreich gelöst
- Erzielte Lösungsqualität
- Laufzeiteffizienz
- Speicherverbrauch
- Rechtfertigung des Lösungsweges
- Antworten auf die theoretischen Fragen der Aufgabenstellung

Voraussetzung für eine positive Absolvierung des Abgabegesprächs ist jedenfalls grundsätzliches Verständnis der Problemstellung.

Zusätzliche Informationen

Lesen Sie bitte auch die auf der Webseite zu dieser LVA veröffentlichten Hinweise. Wenn Sie Fragen oder Probleme haben, wenden Sie sich rechtzeitig an an die AlgoDat2-Hotline unter `algodat2-ss12@ads.tuwien.ac.at`.