RWTH Aachen **SS 2010**
Lehrgebiet Theoretische Informatik Übungsblatt 7
Kneis—Langer—Rossmanith 16.06.2010

Übung zur Vorlesung Effiziente Algorithmen

Tutoraufgabe 16

Gegeben ist folgender Algorithmus $\mathcal{VC}$ für Vertex Cover:

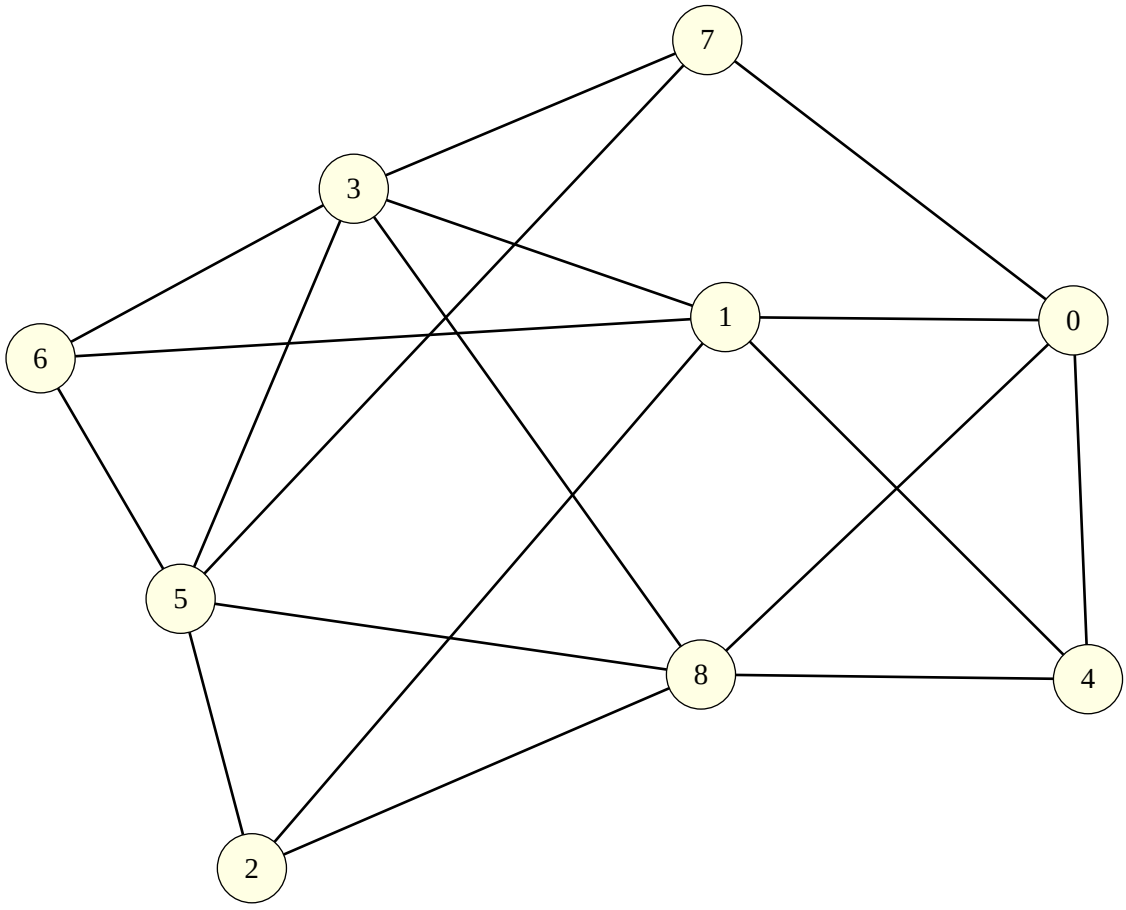
Eingabe: $G = (V, E)$, k

Falls $k \geq 0$ und $E = \emptyset$ return "ja"
Falls $k \leq 0$ return "nein"
Wähle die Kante $\{v_1, v_2\} \in E$ mit kleinster ID.
return $\mathcal{VC}(G \setminus \{v_1\}, k-1)$ or $\mathcal{VC}(G \setminus \{v_2\}, k-1)$

Geben Sie möglichst gute Abschätzungen für die Größe des Suchbaums des Algorithmus auf verschiedenen Graphen an. Die Graphen werden in der Übung zur Verfügung gestellt.

Lösung

Wir betrachten folgenden Zufallsgraphen für $k = 5$:



Die ID einer Kante $\{u, v\}$ ist hierbei einfach das Paar $(ID(u), ID(v))$ falls $ID(u) \leq ID(v)$ und sonst $(ID(v), ID(u))$.

Wir müssen nun den Algorithmus simulierenund dabei eine Verzweigung aus allen Möglichkeiten zufällig auswählen. Seien 0011110100 unsere Zufallsbits. Eine 0 bedeutet, daß wir dem ersten rekursiven Aufruf folgen, eine 1 das wir dem zweiten folgen (da es immer genau zwei Möglichkeiten gibt).

Im ersten Schritt wählen wir also die Kante $\{0, 1\}$ und folgen dem Branch in dem 0 zum Vertex Cover hinzugefügt wird. Wir notieren, daß es zwei Möglichkeiten gab und fahren fort auf der Eingabe $(G \setminus 0, k - 1)$. Die folgende Tabelle gibt die weiteren Schritte an:

Kante	Branch	Möglichkeiten	k	Graph	Gewählte Knoten
$\{0, 1\}$	0	2	4	$G \setminus \{0\}$	$\{0\}$
$\{1, 2\}$	0	2	3	$G \setminus \{0, 1\}$	$\{0, 1\}$
$\{2, 5\}$	1	2	2	$G \setminus \{0, 1, 5\}$	$\{0, 1, 5\}$
$\{2, 8\}$	1	2	1	$G \setminus \{0, 1, 5, 8\}$	$\{0, 1, 5, 8\}$
$\{3, 6\}$	1	2	0	$G \setminus \{0, 1, 5, 8, 6\}$	$\{0, 1, 5, 8, 6\}$

Der Algorithmus bricht nun ab, da die Kante $\{3, 7\}$ noch existiert und $k \leq 0$ gilt. Als Laufzeitschranke erhalten wir $\sum_{i=0}^{5} 2^i = 2^6 - 1$.

Man beachte, daß der von uns zufällig gewählte Pfad zu einem negativen Ergebnis kommt, es aber in dem Graphen sehr wohl ein Vertex Cover der Größe 5 gibt: Im letzten Branch wäre es besser gewesen, Knoten 3 zu wählen. Dies hätte ein Vertex Cover erzeugt ($2, 4, 6, 7$ sind ein Independent Set).

Tutoraufgabe 17

Gegeben ist folgender Algorithmus $\mathcal{DS}$ für Dominating Set:

Eingabe: $G = (V, E)$, k, $V' \subseteq V$

Sei $U = V \setminus N[V']$
Falls $|V'| \leq k$ und $U = \emptyset$ return "ja"
Falls $|V'| \geq k$ return "nein"
Wähle den Knoten $v \in U$ mit kleinster ID.
l="nein";
Für alle Knoten $u \in N[v]$
    Falls $\mathcal{DS}(G, k, V' \cup \{u\})$ setze l="ja"
return l;

Geben Sie möglichst gute Abschätzungen für die Größe des Suchbaums des Algorithmus auf verschiedenen Graphen an. Die Graphen werden in der Übung zur Verfügung gestellt.

Lösung

Wir betrachten wieder den Graphen aus der Lösung zu Tutoraufgabe 16. Analog zur letzten Aufgabe geben wir die rekursiven Aufrufe in einer Tabelle an. Da wir teilweise mehr als zwei Möglichkeiten zur Verzweigung haben, benötigen wir entsprechend viele zufällige Bits (ZB).

v	ZB	u	Möglichkeiten	k	Dominiert	V'
0	10	4	4	1	$\{0,1,4,8\}$	$\{4\}$
2	00	2	4	2	$\{0,1,4,8,2,1,5\}$	$\{4,2\}$
3	100	6	6	3	$\{0,1,4,8,2,1,5,6,3\}$	$\{4,2,6\}$
7	00	0	4	4	$\{0,1,4,8,2,1,5,6,3,7\}$	$\{4,2,6,0\}$

Der Algorithmus bricht nun ab, da alle Knoten dominiert sind. Als Laufzeitschranke erhalten wir

$$1 + 1 \cdot 4 + 4 \cdot 4 + 16 \cdot 6 + 96 \cdot 3 + 288 \cdot 4 = 1557.$$

Tutoraufgabe 18

Geben Sie einen Branch&Bound Algorithmus für folgendes Problem an:

Eingabe: Ein Rechteck R und eine Menge von Rechtecken $R_1, \ldots, R_l$
Frage: Kann man alle Rechtecke $R_1, \ldots, R_l$ ohne Überschneidungen in R achsenparallel platzieren.

Ein Rechteck ist einfach ein Paar $(r_1, r_2) \in \mathbf{N}^2$.

Lösung

Jedes Packet kann entweder horizontal oder vertical plaziert werden. Wir sagen ein eine Platzierung $P(R_i)$ eines Packetes R_i ist ein Tupel (x_1, x_2, o) mit $x1, x_2 \in R$,und $o \in \{h, v\}$. Hierbei gibt (x_1, x_2) die linke untere Ecke des Ortes an, an dem das Packet R_i liegt und o die Orientierung.

Die Menge der Knoten $B(P(R_i))$, die von einem Packet $R_i = (r_1, r_2)$ mit Platzierung $P(R_i) = (x, y, 0)$ belegt werden, ist dann definiert als $\{(a, b) \mid x \leq a \leq x + r_1, y \leq b \leq y + r_2\}$ falls $o = h$ und $\{(a, b) \mid x \leq b \leq x + r_1, y \leq a \leq y + r_2\}$ andernfalls.

Sei $\{P(R_1) \ldots P(R_i)\}$ eine Menge von Platzierungen. Diese ist gültig für R falls $B(P(R_s)) \cap B(P(R_t)) = \emptyset$, für alle $1 \leq s, t \leq i$, $s \neq t$ und gleichzeitig $B(P(R_s)) \in R$. Die erste Bedingung gibt also an, daß Packete sich nicht überlappen, die zweite daß sie komplett in R enthalten sind.

Mit diesen Begriffen können wir nun leicht einen einfachen Backtrackingalgorithmus $\mathcal{B}$ angeben:

Eingabe: R, $R_1, \ldots, R_n$, Platzierungen $P := P(R_1 =, \ldots, P(R_i)$.

Falls $P(R_1), \ldots, P(R_i)$ nicht gültig für R ist**return false**;
Falls $i = n$ **return true**;
Berechne freie Positionen $F = R \setminus \bigcup_{1 \leq s \leq i} B(P(R_s))$;
Für jedes (x, y) in F **do**
    Falls $\mathcal{B}(R, R_1, \ldots, R_n, P(R_1), \ldots, P(R_i), (x, y, v))$ **return true**;
    Falls $\mathcal{B}(R, R_1, \ldots, R_n, P(R_1), \ldots, P(R_i), (x, y, h))$ **return true**;
return false;

Offensichtlich ist der Algorithmus korrekt: Er kann nur dann true zurück geben, falls tatsächlich eine Lösung gefunden wurde, da in Zeile 1 überprüft wird, ob die Packete korrekt platziert sind. Zudem findet er eine Lösung, falls eine existiert, da er für das i-te Packet alle möglichen Platzierungen ausprobiert und sich dann rekursiv selbst aufruft. Formal kann dies leicht per Induktion bewiesen werden.

Um einen Branch&Bound Algorithmus zu erhalten, brauchen wir nun nur noch sinnvolle Stopbedingungen. Solche sind z.B.

- Es gibt ein Packet i, das nirgends mehr platziert werden kann.
- Der Restplatz F ist zu klein, um alle übrigen Packete aufzunehmen.

Diese Abbruchbedingungen sind offensichtlich korrekt.

Tutoraufgabe 19
Geben Sie einen möglichst guten Branch&Bound Algorithmus für das DOMINATING SET Problem an.

Lösung
Wir verwenden den Backtracking Algorithmus aus Aufgabe 17 und müssen somit nur ncoh Abbruchbedingungen angeben:

- Der Durchmesser des Graphen ist mindestens $3k + 1$: Sei P ein kürzester Pfad zwischen zwei Knoten. Offensichtlich ist jeder Knoten $v \notin P$ zu höchstens 3 Knoten aus P adjazent, da er andernfalls ein Shortcut wäre. Jeder Knoten $v \in P$ is ebenso zu höchstens 2 Knoten in P adjazent. Wir brauchen also mindestens $k + 1$ Knoten, umo einen kürzesten Pfad maximaler Länge zu dominieren.
- Es gibt eine Menge von Knoten $V' \subseteq V$ mit $|V'| > k$, sodaß der Abstand $dist(u, v) \geq 3$ für alle $u, v \in V'$. Es ist klar, daß jeder Knoten nur höchstens einen Knoten aus V' dominieren kann.

Man beachte, daß die zweite Bedingung nicht effizient berechnet werden kann. Es ist aber ausreichend, eine solche Menge V' greedy zu berechnen und als Abbruchsbedingung zu benutzen. V' ist dann nicht optimal, aber zumindest inklusions maximal.

Hausaufgabe 14 (10 Punkte)
Geben Sie einen möglichst guten Branch&Bound Algorithmus für das VERTEX COVER Problem an. Beweisen Sie die Korrektheit Ihres Verfahrens.

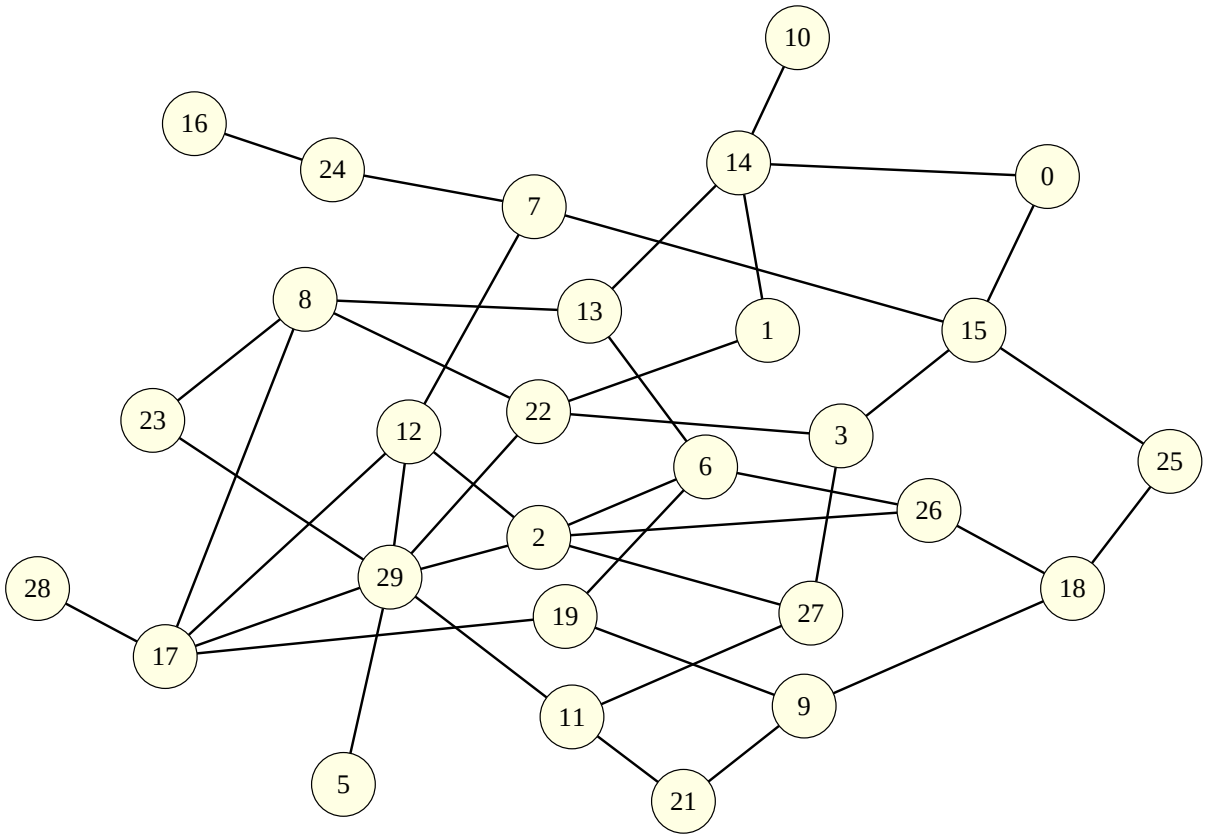
Lösung
Wir verwenden wieder den Algorithmus aus Tutoraufgabe 16. Als Abbruchbedingung berechen wir ein maximales Matching M. Falls dieses Matching mindestens die Größe $k + 1$ hat, brechen wir ab.
Wir müssen nun beweisen, daß diese Abbruchsbedingung korrekt ist. Dazu reicht es aus zu zeigen, daß ein Graph, der ein Matching der Größe $k + 1$ besitzt, kein Vertex Cover der Größe k haben kann.
Sei M also ein Matching der Größe $k + 1$ in G und V' ein Vertex Cover der Größe k in G. Da M ein Matching ist, sind die Endpunkte aller Kanten in M verschieden. Da $|V'| \leq k$, liegen auch höchstens k Endpunkte der Kanten des Matchings in V'. Da M aber aus $k + 1$ Kanten besteht, gibt es somit eine Kante $e \in M$, deren Endpunkte beide nicht in V' liegen. Das bedeutet aber, daß V' kein Vertex Cover ist, ein Widerspruch zur Annahme.

Hausaufgabe 15 (10 Punkte)
Gegeben ist folgender Algorithmus $\mathcal{IS}$ für Independent Set:

Eingabe: $G = (V, E)$, k

Falls $k \leq 0$ return "ja"
Falls $V = \emptyset$ return "nein"
Wähle den Knoten $v \in V$ mit kleinster ID.
return $\mathcal{IS}(G \setminus \{v\}, k)$ or $\mathcal{IS}(G \setminus N[v], k-1)$

Geben Sie eine möglichst gute Abschätzung für die Größe des Suchbaums des Algorithmus auf dem unten angegebenem Graphen für $k = 7$ an.



Lösung
Seien 00010010101111010111100 unsere Zufallsbits. Im ersten Branch wird ein Knoten nicht zum Independent Set hinzugefügt, im zweiten wird er es. Die folgende Tabelle gibt den jeweils den Stand nach dem aktuellen Branch an:

v	Branch	Möglichkeiten	k	Graph
0	0	2	7	$G \setminus \{0\}$
1	0	2	7	$G \setminus \{0, 1\}$
2	0	2	7	$G \setminus \{0, 1, 2, \}$
3	1	2	6	$G \setminus \{0, 1, 2, 3, 15, 22, 27\}$
5	0	2	6	$G \setminus \{0, 1, 2, 3, 15, 22, 27, 5\}$
6	0	2	6	$G \setminus \{0, 1, 2, 3, 15, 22, 27, 5, 6\}$
7	1	2	5	$G \setminus \{0, 1, 2, 3, 15, 22, 27, 5, 6, 7, 12, 15, 24\}$
8	0	2	5	$G \setminus \{0, 1, 2, 3, 15, 22, 27, 5, 6, 7, 12, 15, 24, 8\}$
9	1	2	4	$G \setminus \{0, 1, 2, 3, 15, 22, 27, 5, 6, 7, 12, 15, 24, 8, 9, 18, 19, 21\}$
10	0	2	4	$G \setminus \{0, 1, 2, 3, 15, 22, 27, 5, 6, 7, 12, 15, 24, 8, 9, 18, 19, 21\}$
11	1	2	3	$G \setminus \{0, 1, 2, 3, 15, 22, 27, 5, 6, 7, 12, 15, 24, 8, 9, 18, 19, 21, 11, 29\}$
13	1	2	2	$G \setminus \{0, 1, 2, 3, 15, 22, 27, 5, 6, 7, 12, 15, 24, 8, 9, 18, 19, 21, 11, 29, 13, 14\}$
16	1	2	1	$G \setminus \{0, 1, 2, 3, 15, 22, 27, 5, 6, 7, 12, 15, 24, 8, 9, 18, 19, 21, 11, 29, 13, 14, 16\}$
17	1	2	0	$G \setminus \{0, 1, 2, 3, 15, 22, 27, 5, 6, 7, 12, 15, 24, 8, 9, 18, 19, 21, 11, 29, 13, 14, 16, 17, 28\}$

Danach terminiert der Algorithmus da $k \leq 0$.
Wir erhalten als Abschätzung also $\sum_{i=0}^{14} 2^i = 2^{15} - 1$.