מבני נתונים ואלגוריתמים - תרגול 7

11 בדצמבר 2011

תרגיל

אלגוריתם Warshall-Floyd מוצא את כל המרחקים בגרף ב$O\left(V^3\right)$. שפרו את האלגוריתם כך שימצא את כל המסלולים הקצרים ביותר בגרף.
יותר מכך - תארו מבנה נתונים התופס $O\left(V^2\right)$ זיכרון המקבל שני צמתים ומחזיר את המסלול הקצר ביותר ביניהם.

תזכורת

נניח שהקדקדים הם $\{1, ..., n\}$.
מגדירים d_{ij}^k = אורך המסלול הקצר היותר מi לj שעובר רק בקדקדים $\{1, ..., k\}$.
מאתחלים $d_{ij}^0 = \begin{cases} w(i, j) & (i, j) \in E \\ \infty & \text{else} \end{cases}$, והרקורסיה היא:

$$d_{ij}^{k+1} = \min\left\{d_{ij}^k,\, d_{i(k+1)}^k + d_{(k+1)j}^k\right\}$$

אלגוריתם 1 אלגוריתם Warshall-Floyd

for k=1 to n:
 for i=1 to n:
  for j=1 to n:
   $d_{ij}^k = \min\left\{d_{ij}^{k-1},\, d_{ik}^{k-1} + d_{kj}^{k-1}\right\}$

פתרון

לכל i, j נחזיק מס' קדקד שיסומן בv_{ij}. הוא יהיה קדקד שדרכו עובר המסלול הקצר ביותר מi לj.
תחילה נאתחל null לv_{ij} (כי אין קדקדים במסלול בין i לj).

אלגוריתם 2 אלגוריתם Warshall-Floyd משופר - פתרון התרגיל

for k=1 to n:
 for i=1 to n:
  for j=1 to n:
   if $\left(d_{ik}^{k-1} + d_{kj}^{k-1} < d_{ij}^k\right)$
    $v_{ij} = k$
   $d_{ij}^k = \min\left\{d_{ij}^{k-1},\, d_{ik}^{k-1} + d_{kj}^{k-1}\right\}$

כדי לקבל את המסלול הקצר ביותר נבצע :

get_path(i, j):
 if (v_{ij} =null):
  return $[i, j]$
 else:
  $L_1 \leftarrow$get_path(i, v_{ij})
  $L_2 \leftarrow$get_path(v_{ij}, j)
  return $L_1 + [v_{ij}] + L_2$

קיבלנו מבנה נתונים עם הפעולות הבאות:

- אתחול - מקבל גרף ומריץ Warshall Floyd משופר
- פונק' get_path שמקבלת קדקדים $i,\ j$ ומחזירה את המסלול הקצר ביותר.

סיבוכיות זיכרון:

שומרים d_{ij} וv_{ij} לכל $i,\ j$, לכן הזיכרון הוא $O\left(V^2\right)$. אין טעם להתחשב בk.

סיבוכיות זמן:

כמו Warshall Floyd - $O\left(V^3\right)$.

הוכחת נכונות:

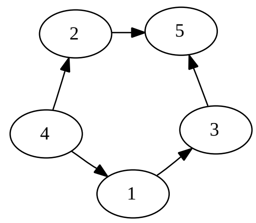
באינדוקציה. באיטרציה הk של הלולאה החיצונית, v_{ij} מכיל קדקד על המסלול הקצר ביותר בין i לj שעובר רק בקדקדים $\{1,\ ...,\ k\}$.
עבור $k=0$, זה נכון כי $v_{ij}=\text{null}$ (כי לא יכולים להיות קדקדים על המסלול בין i לj).
נניח נכונות עבור $k-1$.
יהיו $i,\ j$ קדקדים.
אם $d_{ik}^{k-1}+d_{kj}^{k-1}<d_{ij}^{k-1}$ אז המסלול הקצר ביותר מi לj העובר בקדקדים $\{1,\ ...,\ k\}$ הוא $i\to\ ..\ \to j\to\ ..\ \to k$. (נובע מהנכונות של $W.F$). לכן, k קדקד על המסלול הקצר ביותר בין i לj כדרוש.
אחרת, אז המסלול הקצר ביותר בין i לj בקדקדים $\{1,\ ...,\ k-1\}$ זהה למסלול הקצר ביותר בין i לj העובר בקדקדים $\{1,\ ...,\ k\}$ ולכן לפי הנחת האינדוקציה v_{ij} כבר קדקד על המסלול הקצר ביותר מi לj.

מיון טופולוגי

הגדרה

יהי $G=(V,\ E)$ גרף. מיון טופולוגי של G הוא סדר מלא (לינארי) על V כך שאם $(v,\ u)\in E$ אז $v\le u$.

דוגמה



מיונים טופולוגיים לדוגמא הם:

$$4\ 2\ 1\ 3\ 5$$
$$4\ 1\ 2\ 3\ 5$$
$$4\ 1\ 3\ 2\ 5$$

הערה

בדר"כ יש כמה מיונים טופולוגיים.

תרגיל

יהי G גרף <u>ללא מעגלים</u> (אם בגרף יש מעגל אז לא יכול להיות מיון טופולוגי). כתבו אלגוריתם המוצא מיון טופולוגי של G.

תובנה

אם מv לא יוצאות קשתות אז v איבר מקסימלי בסדר כלשהו.

רעיון

נמצא איבר מקסימלי, נמחק אותו מהגרף, ונחזור על התהליך עד שאין קדקדים בגרף.

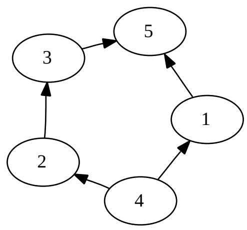
פתרון

אלגוריתם 3 מציאת מיון טופולוגי

נחזיק תור של קדקדים בלי קשתות היוצאות מהם ונעדכן אותו כל פעם כשנמחק קדקד.

Q $\leftarrow$empty queue
for $v \in V$:
 if (num_of_edges_from (v))==0:
  Q.push(v)
if Q.isEmpty:
 error (יש מעגל)
P $\leftarrow$empty queue (זה יהיה תור הפלט)
while Q.isNotEmpty:
 v=Q.pop
 P.push(v)
 for all $(x,\ v)$ $\in$edges_to(v)
  if len(edges_from(x)==1)
   Q.push(x)
G.remove_vertex(v)
end while
if (size(P)¡n):
 error (יש מעגל)
return P

דוגמה



תחילה 5 נכנס לQ, ואז מכניסים אותו לP, מכניסים את 1,3 לQ ומוחקים את 5 (מהגרף ומQ).
אח"כ מכניסים את 1 לP ומוחקים מהגרף ומQ.
אח"כ מכניסים את 3 לP, מכניסים את 2 לQ ומוחקים את 3 מQ ומהגרף.
אח"כ מכניסים את 2 לP, מכניסים את 4 לQ ומוחקים את 2 מQ ומהגרף.
אח"כ מכניסים את 4 לP, מוחקים אותו מQ והגרף וסיימנו, מקבלים:

$$P = \{4,\ 2,\ 3,\ 1,\ 5\}$$

מיון טופולוגי בעזרת DFS

מימוש ספציפי של DFS שנדבר עליו על גרף מכוון:
לקדקד יש 2 צבעים - שחור ולבן. בהתחלה כולם לבנים.

אלגוריתם 4 DFS

```
DFS(V,E):
 Q ←empty queue
 for v∈V
  Q.push(v)
 S ←empty stack
 while Q.isNotEmpty:
  if S.isEmpty
   v ← Q.pop()
   if color(v) is white
    S.push(v)
   else
    continue
  u ← S.top()
  if color(v) is black
   S.pop()
   continue
  else:
   color(v)=black
   for x in verticesFrom(v)
    S.push(x)
```

נקבל יער DFS.

טענה

אם v יצא (בפעם הראשונה) מהמחסנית לפני u ו־G חסר מעגלים אז אין מסלול מ־v ל־u.