20/1/09

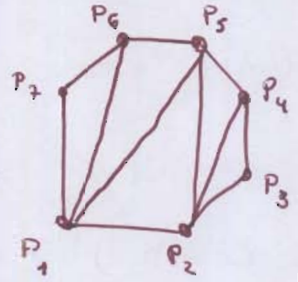
אלגוריתמים - שיעור 12

בשיעור שעבר ראינו דוגמאות רבות לתכנות דינאמי. היום נעבור ל... עוד אחת!

טריאנגולציה עם משקל מינימלי

נתון מצולע קמור P עם n קודקודים* במישור. ע"י העברת מיתרים שלא חוצים זה את זה נפרק את P למשולשים.

(תמיד יהיו בדיוק $n-2$ משולשים. זו חידה חמודה להוכיח למה.)



יש במצולע בסה"כ $\binom{n}{3}$ משולשים פוטנציאליים. נניח שלכל משולש יש משקל $w(\Delta)$. למשל: היקף המשולש. משקל הטריאנגולציה הוא, באופן לא מפתיע, סכום משקלות המשולשים שלה.

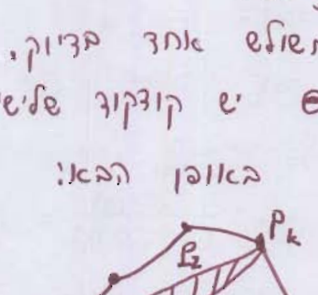
מטרתנו: לבנות טריאנג' עם משקל מינימלי.

נשים לב שעל כל צלע יושב משולש אחד בדיוק.

נסמן $e=P_nP_1$. למשולש שיושב על e יש קודקוד שלישי P_k, עבור $2\le k\le n-1$.

בחירת k מחלקת לנו את המצולע באופן הבא:

ב- P_1 יש k קודקודים.
ב- P_2 יש $n-k+1$.



כעת משקל הטריאנג' המינימלית עבור k זה הוא: $w(P_1P_kP_n)+w(\text{טריאנג' מינ' של } P_1)+w(\text{טריאנג' מינ' של } P_2)$. לכן, גישה רקורסיבית היא לחפש את k עבורו ערך זה מינימלי.

נפצל כל תת-מצולע במיתר שממנו קיבלנו אותו, וזה יבטיח שכל תת-מצולע לאורך כל הרקורסיה יהיה מורכב מחלק רציף משפת P + מיתר. נתאר כל תת-בעיה P' ע"י (P_i,P_j) והכוונה היא לזה ששפתו היא מ- P_i עד P_j נגד כיוון השעון (+ המיתר P_iP_j). סה"כ יש לנו $O(n^2)$ תת-מצולעים אפשריים.

אם $j=i+1$ זה מצולע ריק שהוא רק צלע. במקרה זה נאמר שמשקלו 0.

אם $j=i+2$ התת-מצולע הוא משולש.

* זאת הפעם הראשונה בקורס שמישהו אומר "קודקודים"!

$$W_{1n} = \min_{2\le k\le n-1}\left[w(p_1p_kp_n) + W_{1k} + W_{kn}\right]$$

$$W_{ij} = \min_{i+1\le k\le j-1}\left[w(p_ip_kp_j) + W_{ik} + W_{kj}\right]$$

כאשר $k-i,\ j-k < j-i$

מכאן כבר ברור כיצד לחשב את ערכי W_{ij} בשיטת bottom-up, דהיינו תכנות דינאמי: נפתור בעיות עם הפרש 1: עבור $i=1,\dots,n-1$, נבצע: $W_{i,i+1} \leftarrow 0$

לולאה על ההפרש (j): עבור $i=1,\dots,n-j$:

$$W_{i,i+j} = \min_{i+1\le k\le j-1}\left[w(p_ip_kp_j) + \underbrace{W_{ik} + W_{kj}}_{\text{הפרשים} < j}\right]$$

סה"כ זמן הריצה $O(n^3)$.

ועכשיו, שוב, למשהו אחר לגמרי, ומלהיב לא פחות.

התאמת מחרוזות — String Matching / Pattern Matching

הקלט: מחרוזת טקסט T (למשל: התנ"ך)
מחרוזת תבנית (Pattern) P (למשל "מאה")

המטרה: למצוא את כל המקומות ב-T בהם P מופיע.

סימונים: $n=|T|$ $\quad T = T_1T_2T_3\dots T_n = T[1]T[2]T[3]\dots T[n]$

$m=|P|$ $\quad P = P_1P_2P_3\dots P_m = P[1]P[2]P[3]\dots P[m]$

הגדרה: P מופיע במקום ה-i של T אם

$T[i]=P[1]$
$T[i+1]=P[2]$
$\vdots$
$T[i+m-1]=P[m]$

דוגמה: $T=$ ABRACADABRA $\quad P=$ ABRA

כל הופעה של P ב-T קורית בהזזה מסויימת (shift), שהיא אינדקס האות של T שמתלכדת עם האות הראשונה של P. בדוגמה שלנו, ההזזות הרלוונטיות הן 1,8.

פתרון נאיבי: לנסות את כל ההזזות האפשריות בזו אחר זו.

(הערה: מותרת חפיפה חלקית של הופעות P ב-T).

זמן ריצה של הפתרון הנאיבי: $O(mn)$.

נניח ש-T מורכב רק מ-2 אותיות A,B באופן אקראי: במקרה זה, בהזזה מסויימת של P נבצע תמיד השוואה ראשונה, בהסתברות $\frac{1}{2}$ נבצע השוואה שנייה,

בהסתברות $\frac{1}{4}$ נבצע השוואה שלישית וכו'. לכן, תוחלת מס' ההשוואות (בתבנית אחת) הוא 2.

במקרה קיצוני $P=AA..A$, $T=AAA...A$. כאן האלגוריתם הנאיבי יבצע $(n-m+1)m$ השוואות, שזה כמעט mn. למרות שעבור אותו P, אם אנחנו יודעים ש-P מופיע במקום ה-i, אין צורך לעשות את אותו מספר השוואות - מספיק לבדוק את התו האחרון. לעומת זאת, אם $P=ABCD$, ואנחנו מוצאים התאמה במקום ה-i, אנחנו יכולים לקפוץ למקום ה- $i+4$ ב-T לצורך החיפוש הבא. היינו רוצים אלגוריתם "חכם" שינצל תבונות כאלה של P.

הגדרה: רישא של P היא i התווים הראשונים של P, עבור איזה i.
באופן דומה, סיפא היא i התווים האחרונים, עבור איזה i.

נניח כי $P=ABABC$, ומצאנו התאמה חלקית, במובן זה שבתבנית 1, יש ב-T התאמה לרישא בגודל 4 של P. אנו יודעים שההצבה הבאה שצריך לנסות היא $i+2$.

הכלל הוא כזה: אם בתבנית מס' i הצלחנו להתאים רישא של P $P[1...i]$, נחפש תבנית מינימלית שבה יש התלכדות בין רישא של $P[1...i]$ לסיפא של אותה רישא.

T = AB|AB|ABC
P = AB|AB|C
  |AB|ABC

במילים אחרות, אני רוצים להכין טבלה כך שלכל רישא $P[1...i]$ של P נדע מהו גודל הרישא המקסימלית של $P[1...i]$ שהיא גם סיפא של $P[1...i]$. $i=1,2,3,...,m$

דוגמה:

$P=ABABACA$

	i	גודל הרישא המינימלית שהיא גם סיפא	גודל התזוזה הנוספת
A	1	0	1
AB	2	0	2
ABA	3	1	2
ABAB	4	2	2
ABABA	5	3	2
ABABAC	6	0	6
ABABACA	7	1	6

לסיכום, נתאר את התהליך. נניח שכבר חישבנו את העמודה האמצעית בטבלה (שלה נקרא $\pi[i]$), ואז נבצע התאמה:

נעבור על T:

for i = 1 to n do

נתחזק אינדקס q = כמה תווים של P כבר מותאמים בהזזה הנוכחית.

נבצע השוואה של $(T[i], P[q+1])$, $(T[i+1], P[q+2])$, ... עד שנכשל או נגיע לסוף P.

אם הגענו לסוף P נודיע על התאמה, ונתנהג כאילו "נכשלנו" אחרי m תווים.

ואז הרישא המקסימלית שהותאמה היא כל P.

נניח k = אורך הרישא שהתאמנו. נחשב את $\pi[k]$ מתוך הטבלה ונבצע: $q \leftarrow \pi[k]$.