שם: רן אלגאוי

ת.ז: 305243115

ממ"ן 14

שאלה 1

נגדיר את OPT[i,j] כפתרון האופטימלי למסלול המנימלי שמתחיל בנקודה כלשהי בשכבה השמאלית ומגיע לי נקודה i,j.

בצורה יותר פורמלית:

נסמן את המטריצה שניתנה לנו באות M, ונוסיף לה מספר שורות/עמודות בצורה הבאה:

* שורה אחת מלמעלה מלאה ב”inf” או כל ערך מוסכם.
* שורה אחת מלמטה מלאה ב"inf" או כל ערך מוסכם.
* עמודה מצד ימין מלאה ב”inf” או כל ערך מוסכם.

נשתמש בשורות האלו בעת ריצת האלגוריתם על מנת לבצע חיושבים.

לפי בקשת התרגיל עלינו להחזיר את OPT[n,j] כאשר j הוא מינימלי ביותר.

אלגוריתם:

1. For i in range 1 to n: // Fill new colums/rows with value inf
   1. for j in range 0 to n+1:
      1. OPT[i,j] = inf
2. For j in range 1 to n:
   1. OPT[i,j] = C(1,j)
3. For i in range 2 to n:
   1. For j in range 1 to n:
      1. OPT[I,j] =
4. min\_price = min(OPT[n,j])
5. for j in range 1 to n:
   1. if OPT[n,j] = min\_price:
      1. jk = j
6. for k in range n to 1:
   1. jk-1 = jk
   2. step\_diagnoal\_down\_is\_optimal = (OPT[k-1, jk-1] + c(k,j­k) == OPT[k,jk])
   3. step\_diagonal\_up\_is\_optimal = (OPT[k-1, jk+1] + c(k,j­k) == OPT[k,jk])
   4. if step\_diagnoal\_down\_is\_optimal:
      1. jk-1 = jk-1
   5. else if step\_diagonal\_up\_is\_optimal:
      1. jk-1 = jk + 1
   6. else: // path goes straight
      1. j­­k-1 = jk

חישוב סיבוכיות:

* שורות 1 עד 3: מילוי תאים בזמן של O(n\*(n+2))=O(n2).
* שורה 4: מציאת מחיר מינימלי מבין כל ערכי n בזמן של O(n).
* שורה 5: סקירה של n אינדקסים והשוואה למחיר המינימלי שמצאנו בשורה 4 בזמן של O(n).
* שורה 6: ביצוע מספר קבוע של פעולות בn איטרציות למצוא מסלול בזמן של O(n).
* סה"כ: O(n2)

שאלה 2

אלגוריתם + הסברים:

1. נמיין את התיבות לפי הרוחב שלהן.
2. נגדיר 2 מערכי עזר בגודל n כך:
   1. מערך OPT - שיכיל את הגודל המרבי של המגדל כך שתיבה i בראשו.
   2. רשימה מקושרת דו-כיוונית עם גישה מיידית כמו מערך TOWER - בכל פעם שנחשב את Opt[i] נשמור בTOWER.prv = j
3. עבור i מ1 עד n:
   1. עבור j מ1 עד n: // עבור ריצה דו מימדית
      1. אם : // נשווה בין הגודל המקסימלי של תיבה i + הגובה של המגדל עבור j. אם נכון קיבלנו את הגובה המקסימלי עבור תיבה i.
         1. TOWER[i].next = null
      2. אחרת: // מצאנו את התיבה את התיבה הקודמת של i
         1. TOWER[i].prv = j
         2. TOWER[j].next = null
4. כל עוד TOWER[s].prv ≠ null:
   1. נדפיס את TOWER[s].prv // נדפיס את כל התיבות במגדל.
   2. s = TOWER[s].prv

הוכחות נוכנות:

* נחשב את הOPT, ביצוע החישוב מתבצע בצורה רקורסיבית לחישוב הגובה המקסימלי של המגדל שהתיבה i בפסגה.
* התיבות שהן אופציה להיות מתחת לתיבה i הן תיבות שקטנות ממנה באורך ורוחב על מנת לשמור יציבות המגדל.
* על ידי שמירה של prv וnxt נוכל לשחזר את המגדל בכל זמן.

חישוב סיבוכיות:

שורה 1: מיון איברים לפי הרוחב שלהם מתבצע בזמן סטנדרטי של O(n\*logn).

שורה 2: יצירת מערך ורשימה מקושרת בגודל הקלט מתבצע בזמן לינארי של O(n).

שורה 3: הכנסה למערך OPT, נצטרך למצוא את הערך המקסימלי לפי אורך ורוחב ולכן לכל היותר בגודל O(n2).

שורה 4: מעבר על לכל היותר n איברים במעבר על TOWER מתבצע בזמן לינארי של O(n).

סה"כ: O(n2)

שאלה 4

סעיף א'

האלגוריתם מחשב את המרחק הקצר ביותר מצומת r אל שאר הצמתים בגרף, כאשר באיבר A[v] נמצע המרחק המינימלי בין הצומת v לצומת r.

הוכחה על ידי אינדוקציה

בסיס האינדוקציה: כאשר i=0, מתקיים A[r]=0, ובאמת המרחק המינימלי בין הצומת r לעצמו ה0.

הנחת האינדוקציה: כאשר i=n, כל הצמתים במרחק n-י מצומת r מעודכנים במערך A.

צעד האינדוקציה:

1. נניח עבור n-1 ונראה נכונות עבור n.
2. נסמן ב-d(u,v) כערך המינימלי בין הצמתים u ו-v.
3. נסמן ב-e(u,v) כקשת בין u ל-v.
4. יהי צומת v כלשהו במרחק n-י מהצומת הנתון r.
5. מכיוון ש-v במרחק n-י מצומת r, הרי שקיים צומת u כלשהו במרחק n-1 צמתים מצומת r שדרכו הגענו לv.
6. לפי הנחת האינדוקציה ברור לנו שהערך A[u] הוא מינימלי ונכון.
7. לפי סעיף 6, A[v] = A[u] + e(u,v) = d(r,u).

סעיף ב'

לפי המסקנה מסעיף א', בכל איטריציה i כלשהי אנחנו מטפלים בכל הצמתים שנמצאים במרחק i-י מצומת r הנתונה ובכל שלב האלגוריתם יטפל בלפחות צומת יחיד ולכן על למנת להגיע לB(n)=n נצטרך לטפל לכל היותר בצומת אחד בכל איטרציה.

דרך קלה להשיג את המסקנה הנ"ל נסדר בסדר לקסיקוגרפי הפוך בצורת שרשרת כך שכל צומת יוביל אותנו אל הצומת בסדר הבא שלנו.

v

r

d(n-1,n) = n-1

….

d(r,v) = 0

d(v,u) = 1