ריכוז אלגוריתמים

מיונים:

שם: **Insertion sort**

מטרה: מיון סדרה בסדר לא יורד

תיאור האלגוריתם : יהא $A = (A_1, \ldots, A_n)$ וקטור של ממשיים שונים. המיון מתבצע ב־ m שלבים, כאשר בסוף השלב ה־ i הסדרה החלקית $A(1), \ldots, A(i)$ ממויינת. בשלב ה־ $(i+1)$ משווים את $A(i+1)$ בזה אחר זה ל־ $\ldots, A(i-1), A(i)$ עד שמוצאים את מקומו ביניהם.

אלגוריתם :

$IS(A, n)$
For $i = 2$ to n
$k \leftarrow A(i)$
$j \leftarrow i - 1$
while $(k < A(j)$ & $j \geq 1)$
    $A(j+1) \leftarrow A(j)$
    $j \leftarrow j - 1$
$A(j+1) \leftarrow k$

סיבוכיות:

$\binom{n}{2}$ או עם חיפוש בינארי : $c_{BIS}(n) \leq \sum_{i=1}^{n} \lceil \log_2 n \rceil \leq n \lceil \log_2 n \rceil$

שם: ***Merge***

מטרה: מיזוג שתי סדרות ממוינות לסדרה אחת

תיאור האלגוריתם : <u>בעיית המיזוג:</u> נתונים שני וקטורים ממויינים זרים $a = (a_1 < \ldots < a_k)$ ו־ $b = (b_1 < \ldots < b_l)$ עלינו למיין את הסידרה $c = \{a_i\}_{i=1}^{k} \cup \{b_j\}_{j=1}^{l}$ כך ש: $c = (c_1 < \ldots < c_{k+l})$.
להלן אלגוריתם השוואת רקורסיבי הממזג את $a \cup b$ לתוך c.

אלגוריתם :

$merge(a, k, l, b, c)$
If $k = l = 0$ stop.
If $k = 0$ do $c(j) \leftarrow b(j),\ j = 1, \ldots, l$ and stop.
If $l = 0$ do $c(i) \leftarrow b(i),\ j = 1, \ldots, k$ and stop.
If $a(k) < b(l)$
    $c(k+l) \leftarrow b(l)$
    $merge(a, k, b, l-1, c)$
If $a(k) > b(l)$
    $c(k+l) \leftarrow a(k)$
    $merge(a, k-1, b, l, c)$

סיבוכיות:

$$\left\lceil \log_2 \binom{k+l}{k} \right\rceil \leq C_{merge}(k, l) \leq k + l - 1$$

$$C_{merge}(k, 1) = \lceil \log_2 (k+1) \rceil$$

$$C_{merge}(k, k) = 2k - 1.$$

שם: **Binary Search**

מטרה: מציאת המיקום של איבר b בתוך סדרה ממוינת A

תיאור האלגוריתם : בהינתן $A = (a_1 < \ldots < a_k)$ ו־ b עלינו למצוא $0 \le i \le k$ עבורו $a_i < b < a_{i+1}$ (כאשר מגדירים $a_0 = -\infty$ ו־ $a_{k+1} = \infty$).

אלגוריתם :

$BS(A, k, b, i)$
If $k = 1$ & $A(1) < b$ then $i = 1$, stop.
If $k = 1$ & $A(1) > b$ then $i = 0$, stop.
If $b > A(\lceil \frac{k}{2} \rceil)$
$A' = (A(\lceil \frac{k}{2} \rceil + 1), \ldots, A(k))$
$BS(A', \lfloor \frac{k}{2} \rfloor, b, i)$
$i \leftarrow i + \left\lceil \frac{k}{2} \right\rceil$
If $b < A(\lceil \frac{k}{2} \rceil)$
$A'' = (A(1), \ldots, \lceil \frac{k}{2} \rceil - 1))$
$BS(A'', \lceil \frac{k}{2} \rceil - 1, b, i)$

סיבוכיות: $c_{BS}(k) \le \lceil \log_2 (k+1) \rceil$

שם: **Merge Sort**

מטרה: מיון סדרה בסדר לא יורד

תיאור האלגוריתם : בכל שלב נפצל את הבעיה לקבוצה קטנה יותר (חלקי 2), ונמיין באמצעות מיזוג.

אלגוריתם :

$MS(A, n)$
If $n = 1$ stop.
$B \leftarrow (A(1), \ldots, A(\lfloor \frac{n}{2} \rfloor)$
$C \leftarrow (A(\lfloor \frac{n}{2} \rfloor + 1), \ldots, A(n))$
$MS(B, \lfloor \frac{n}{2} \rfloor)$
$MS(C, \lceil \frac{n}{2} \rceil)$
$merge(B, \lfloor \frac{n}{2} \rfloor, C, \lceil \frac{n}{2} \rceil, A)$

סיבוכיות: $c_{MS}(n) \le n \lceil \log_2 n \rceil \le n \log_2 n + n.$

אלגוריתמים מהתרגול:

שם	מטרה	תיאור האלגוריתם	סיבוכיות:
QUICK-SORT	מיון סדרה בסדר לא יורד	כמו Merge-Sort רק במקום לפצל לשני מערכים זהים בגודלם, נפצל לפי pivot שנבחר רנדומלי.	במקרה הסתברותי - O(n log) במקרה הגרוע - $O(n^2)$

```
QUICK-SORT(A, p, r)
  if p < r
     then q ← PARTITION(A, p, r)
          QUICK-SORT(A, p, q)
          QUICK-SORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
  x ← A[p]
  i ← p − 1
  j ← r + 1
  while true
      do repeat j ← j − 1
            until A[j] ≤ x
         repeat i ← i + 1
            until A[i] ≥ x
         if i < j
            then exchange A[i] ↔ A[j]
            else return j
```

שם	מטרה	תיאור האלגוריתם	סיבוכיות:
COUNTING-SORT	מיון סדרה בסדר לא יורד	כאשר ידוע לנו טווח המספרים, נעשה היסטוגרמה במערך חדש, ואז נכניס למערך כמספר הפעמים שכל איבר מופיע.	O(n + k)

```
COUNTING-SORT(A, B, k)
  for i ← 1 to k
      do C[i] ← 0
  for j ← 1 to length[A]
      do C[A[j]] ← C[A[j]] + 1
  ▷ C[i] is the number of times i appears as an entry in A.
  for i ← 2 to k
      do C[i] ← C[i] + C[i − 1]
  ▷ C[i] is the now the number of elements ≤ i appearing in A.
  for j ← length[A] downto 1
      do B[C[A[j]]] ← A[j]
         C[A[j]] ← C[A[j]] − 1
```

שם	מטרה	תיאור האלגוריתם	סיבוכיות:
BUCKET-SORT	מיון סדרה בסדר לא יורד	בהנחה שהפיזור אחיד, וידוע לנו טווח המספרים- נמיין את המספרים לN קבוצות זרות ושוות בגודלן, מיון כל קבוצה הוא סופי	O(n log n)

```
BUCKET-SORT(A)
  ▷ We divide the allowed range into n equally sized consecutive buckets.
  for i ← 1 to length[A]
      do Place A[i] into the appropriate bucket.
  Sort each one of the n buckets (using heap or merge sort).
  Output the (sorted) contents of each bucket according to the order of the buckets.
```

שם	מטרה	תיאור האלגוריתם	סיבוכיות:
RADIX-SORT	מיון סדרה בסדר לא יורד	במיון זה, אנו מסתמכים על כך שידוע לנו מספר הספרות בכל מספר והוא קבוע, מיון זה הוא יציב- ממיינים לפי ספרת האחדות, לאחר מכן העשרות וכו' כאשר שומרים על הסדר מהמיון הקודם בכל שלב	O(d(n + k))

```
RADIX-SORT(A, d)
  for i ← 1 to d
      do Apply a stable sorting algorithm to sort A according to the digit i.
```

<u>ערימות:</u>

שם: Heapify

מטרה: בהנחה ששני תתי העצים הם ערימות, להפוך את העץ כולו לערימה

תיאור האלגוריתם : יהא T עץ בינארי מאוזן על קבוצת הקודקודים V, $|V| = n$ ויהא $A = (A(v))_{v\in V}$ מערך המאונדקס על ידי V. A ייקרא <u>ערימה</u> אם לכל $v \in V$ מתקיים

$$A(v) \geq \max\{A(left(v)), A(right(v))\}.$$

אלגוריתם :

Heapify(A, n, v)
if $A(v) \geq \max\{A(left(v)), A(right(v))\}$ stop.
else if $A(left(v)) > A(v)$

exchange $A(left(v))$ & $A(v)$
$Heapify(A, n, left(v))$

else

exchange $A(right(v))$ & $A(v)$
$Heapify(A, n, right(v))$

סיבוכיות: $O(height(v))$

שם: Make Heapify

מטרה: הפיכת מערך לערימה

תיאור האלגוריתם : יוצאים מנקודת הנחה שהבנים הם כבר ערימות, דרך האלגוריתם היא "פעפע את A(v) למטה"

אלגוריתם :

makeheap(A,n)
For $i = n$ down to 1
$Heapify(A, n, i)$

סיבוכיות: $O(n)$

שם: Heapsort

מטרה: מיון באמצעות ערימה

תיאור האלגוריתם : יוצרים ערימה ממערך באמצעות make heap. בשלב הבא מחליפים בין האיבר המקסימלי- השורש, לבין האיבר האחרון בערימה, ומוציאים את המקסימלי חזרה למערך, מתקנים במידת הצורך את הערימה.

אלגוריתם :

<u>$Heapsort(A, n)$</u>

$makeheap(A, n)$
For $i = n$ down to 2
exchange $A(1) \leftrightarrow A(i)$
$Heapify(\{A(1), \ldots, A(i-1)\}, 1)$

סיבוכיות: $O(n \log n)$

אלגוריתם בחירה SELECT:

מטרה: סדרה לא ממוינת, למצוא את האיבר ה-k באותה סדרה אם הייתה ממוינת

אלגוריתם : $\underline{Sel(A,n,k)}$

(א) נחלק את A לחמישיות $F_1,\dots,F_{\frac{n}{5}}$ ונמצא את החציון של כל חמישייה:

$$b_i \leftarrow Sel(F_i,5,3)$$

יהא $B=(b_1,\dots,b_{\frac{n}{5}})$ וקטור החציונים.

(ב) יהא $x \leftarrow Sel(B,\frac{n}{5},\frac{n}{10})$ החציון של B.

(ג) נחשב:

$$C=\{a_i\in A : a_i<x\}$$

$$D=\{a_i\in A : a_i>x\}$$

(ד)

$$Sel(A,n,k)\leftarrow \begin{cases} Sel(C,|C|,k) & |C|\geq k \\ x & |C|=k-1 \\ Sel(D,|D|,k-|C|-1) & |C|\leq k-2 \end{cases}$$

סיבוכיות:

- $C_{sel}(n,1)=n-1$
- $C_{sel}(n,2)=n+\lceil\log_2 n\rceil-2$

$$c_{Sel}(n,k)\leq \widetilde{S}(n)\leq \frac{3n}{1-\frac{1}{5}-\frac{7}{10}}=30n$$

בעיית הקידוד חסר הרעש –

משפט Shanon - $H(p) \leq f(p) \leq H(p) + 1$, כאשר $H(p)$ היא פונקציית האנטרופיה המוגדרת על וקטור הסתברויות: $H(p) = H(p_1, \dots, p_M) = \sum_{i=1}^{M} p_i \cdot \log_2 \frac{1}{p_i}$.

אלגוריתם Huffman למציאת $f(p_1, \dots, p_M)$

מטרה: מציאת קוד רישא אופטימלי לווקטור הסתברויות מסוים כאשר עלינו לקודד קובץ עם n אותיות כך שאפשר יהיה לשחזר את הקובץ המקורי מהקובץ המקודד, ושהקובץ המקודד יהיה מינימלי (=מציאת צופן רישא).

אלגוריתם :

אלגוריתם הופמן למציאת קוד רישא אופטימלי ל־ $p = (p_1, \dots, p_M)$:
מצא את שני האיברים המינימליים ב־ p. בלי הגבלת הכלליות $p_1, \dots, p_{M-2} \geq p_{M-1}, p_M$.
מצא ברקורסיה קוד רישא אופטימלי ל־ $q = (p_1, \dots, p_{M-2}, p_{M-1} + p_M)$ ופצל את הקודקוד המתאים ל־ $p_{M+1} + p_M$ לשני בניו.

עצים פורשים מינימליים והאלגוריתם החמדן –

טענה (תכונת החילוף ליערות): יהיו $F_1 = (V, E_1)$, $F_2 = (V, E_2)$ שני יערות על אותה קבוצת קודקודים V. אם $|E_1| < |E_2|$ אזי קיימת צלע $e \in E_2 - E_1$ כך ש־ $(V, E_1 \cup \{e\})$ יער.

אלגוריתם Kruskal למציאת עץ פורש מינימלי

מטרה: מציאת עץ פורש מינימלי

אלגוריתם : בכל שלב מוצאים את הצלע בעלת המשקל הנמוך ביותר כל עוד היא לא יוצרת מעגל, מסמנים אותה. ממשיכים באותה צורה עד שלא ניתן להוסיף יותר צלעות. הצורה שהתקבלה היא עץ פורש מינימלי.

סיבוכיות: $O(|E| \log|E|)$

אלגוריתם PRIM למציאת עץ פורש מינימלי

מטרה: מציאת עץ פורש מינימלי

אלגוריתם : בכל שלב אנו שומרים על הגרף שאנחנו יוצרים כעץ (כלומר קשיר). בכל שלב מוצאים את הצלע בעלת המשקל הנמוך ביותר שמחוברת באחד מקודקודיה לצלע שנבחרה כבר (כל עוד היא לא יוצרת מעגל), מסמנים אותה. ממשיכים באותה צורה עד שלא ניתן להוסיף יותר צלעות. הצורה שהתקבלה היא עץ פורש מינימלי.

סיבוכיות: $O(|E| \log|V|)$

אלגוריתם BFS חיפוש רוחב

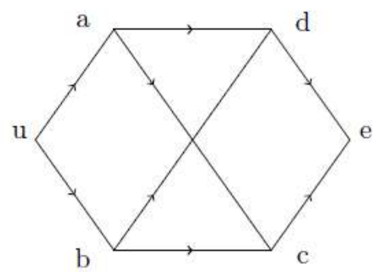
מטרה: בהינתן גרף מכוון $G = (V, E)$ על n קודקודים, וקודקוד קבוע $u \in V$, לסרוק את כל קודקודי V החל מ-u.

תיאור האלגוריתם : בכל שלב באלגוריתם יש שתי סדרות של קודקודים: S, R. S = סדקת קודקודים שכבר טופלו. R = סדרת קודקודים שעדיין בטיפול. נסמן $d*$(v) – המרחק של v מהקודקוד u. נסמן $\pi(v)$ – הקודקוד האחרון שביקרנו בו כדי להגיע לקודקוד v מהקודקוד u.

אלגוריתם: <u>איטרציה:</u> יהא v הקודקוד השמאלי ב־ R. אם קיים $x \in \Gamma(v) - (S \cup R)$ הוסף את x ל־ R מימין. אם $\Gamma(v) \subset S \cup R$, השמט את v מ־ R והוסף אותו מימין ל־ S. כאשר $R = \emptyset$ סיים וסמן $S = (u = v_1, v_2, \ldots, v_n)$.
אלגוריתם BFS מאפשר חישוב פונקציות שונות על G.

סיבוכיות: $O(|E| + |V|)$

דוגמת הרצה: <u>דוגמא:</u>

$G =$ 

בכל שלב באלגוריתם נסמן $(d^*(v), \pi(v))$ מתחת לקודקוד v שהתגלה.

S	R	u	a	b	c	d	e
$\emptyset$	u	$(0, \emptyset)$					
$\emptyset$	ua	"	$(1, u)$				
$\emptyset$	uab	"	"	$(1, u)$			
u	ab	"	"	"			
u	abc	"	"	"	$(2, a)$		
u	$abcd$	"	"	"	"	$(2, a)$	
ua	bcd	"	"	"	"	"	
uab	cd	"	"	"	"	"	
uab	cde	"	"	"	"	"	$(3, c)$
$uabcde$	$\emptyset$	$(0, \emptyset)$	$(1, u)$	$(1, u)$	$(2, a)$	$(2, a)$	$(3, c)$

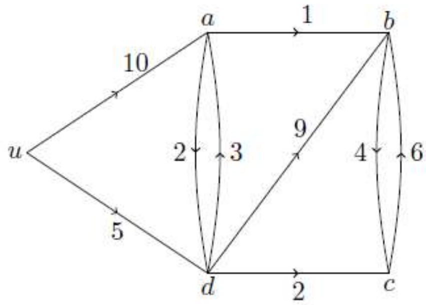
<u>אלגוריתם Dijkstra</u>

מטרה: מציאת המסלול הקצר ביותר בין כל שני קודקודים בגרף.

אלגוריתם : האלגוריתם עובד על גרף נתון, לאו דווקא מכוון, בעל משקלות אי- שליליות על הצלעות בגרף המסמלות מרחק. בתחילה מסמנים את כל המרחקים מהמקור לקודקודים כ-∞. מסמנים בקו תחתון את הקודקוד שביקרנו בו. עבור כל קודקוד שהוא שכן של X, נעדכן את ערכו של המרחק בין הערך המינימלי בין ערכו הנוכחי לבין משקל הקשת שמחברת בין X ל-Y בתוספת המרחק בין S ל-X. בכל שלב בוחרים את הקודקוד הלא מסומן בעל המרחק הקצר ביותר. אותו מרחק הוא המינימלי (בגלל אי שליליות המשקלות). במידה ועדכנו את המרחק בטבלה של d, נעדכן את הקודקוד האחרון שהיינו בו, בטבלה של π.

סיבוכיות: $O(|V|^2)$

דוגמת הרצה:



d_k

u	a	b	c	d
$\underline{0}$	10	∞	∞	5
$\underline{0}$	8	14	7	$\underline{5}$
$\underline{0}$	8	13	$\underline{7}$	$\underline{5}$
$\underline{0}$	$\underline{8}$	9	$\underline{7}$	$\underline{5}$
$\underline{0}$	$\underline{8}$	$\underline{9}$	$\underline{7}$	$\underline{5}$

π_k

u	a	b	c	d
$\underline{\emptyset}$	u	u	u	u
$\underline{\emptyset}$	d	d	d	$\underline{u}$
$\underline{\emptyset}$	d	c	$\underline{d}$	$\underline{u}$
$\underline{\emptyset}$	$\underline{d}$	a	$\underline{d}$	$\underline{u}$
$\underline{\emptyset}$	$\underline{d}$	$\underline{a}$	$\underline{d}$	$\underline{u}$

למשל, המסלול המינימלי מ־ u ל־ b הוא

$$u = \pi_5(d) \to d = \pi_5(a) \to a = \pi_5(b) \to b$$

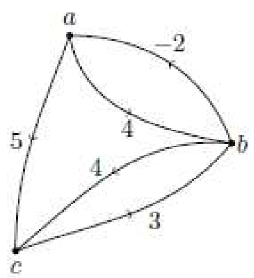
אלגוריתם Floyd-Warshall

מטרה: לחשב את $d(i,j) = \inf\{w(P) : P \text{ מסלול מכוון מ־} i \text{ ל־} j\}$

אלגוריתם : האלגוריתם עובד על גרף מכוון, בעל משקלות ($\in R$) על הצלעות בגרף המסמלות מרחק. בתחילה מסמנים את כל המרחקים מהמקור לקודקודים כ-∞. ממספרים את הקודקודים מ-1 עד k. עוברים כל פעם על קודקוד אחר ובודקים האם מעבר דרכו מקצר את המרחק מקודקוד המקור לכל קודקוד אחר - $d_k(x,y) = \min\{d_{k-1}(x,y), d_{k-1}(x,k) + d_{k-1}(k,y)\}$. במידה וכן, מעדכנים את המרחק ואת הקודקוד האחרון שעברנו בו בטבלה של ה-π_i הנוכחי.

סיבוכיות: $O(|V|^3)$

דוגמת הרצה:



$$D_0 = \begin{array}{c|ccc|} & a & b & c \\ \hline a & 0 & 4 & 5 \\ b & -2 & 0 & 4 \\ c & \infty & 3 & 0 \\ \hline \end{array} \qquad \pi_0 = \begin{array}{c|ccc|} & a & b & c \\ \hline a & \emptyset & a & a \\ b & b & \emptyset & b \\ c & \emptyset & c & \emptyset \\ \hline \end{array}$$

$$D_1 = \begin{array}{|ccc|} \hline 0 & 4 & 5 \\ -2 & 0 & 3 \\ \infty & 3 & 0 \\ \hline \end{array} \qquad \pi_1 = \begin{array}{|ccc|} \hline \emptyset & a & a \\ b & \emptyset & a \\ \emptyset & c & \emptyset \\ \hline \end{array}$$

$$D_2 = \begin{array}{|ccc|} \hline 0 & 4 & 5 \\ -2 & 0 & 3 \\ 1 & 3 & 0 \\ \hline \end{array} \qquad \pi_2 = \begin{array}{|ccc|} \hline \emptyset & a & a \\ b & \emptyset & a \\ b & c & \emptyset \\ \hline \end{array}$$

$$D_3 = D_2 \qquad \pi_3 = \pi_2$$

מסלול מינימלי מ־b ל־c:

$$b = \pi_3(b,a) \to a = \pi_3(b,c) \to c.$$

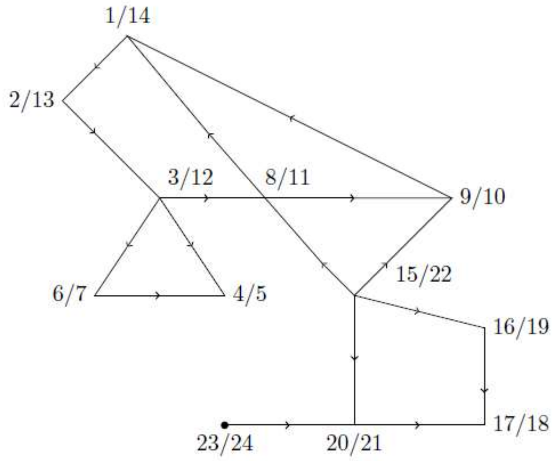
<u>אלגוריתם DFS חיפוש עומק</u>

מטרה: לחשב את $d(i,j) = \inf\{w(P) : P$ מסלול מכוון מ־i ל־$j\}$

תיאור האלגוריתם : האלגוריתם מתחיל את החיפוש מצומת שרירותי בגרף, ומתקדם לאורך הגרף עד שהוא נתקע, לאחר מכן הוא חוזר על עקבותיו עד שהוא יכול לבחור להתקדם לצומת אליו טרם הגיע. דרך הפעולה היא רקורסיבית, מתחילים בקודקוד המקור ומפעילים אותו רקורסיבית על כל אחד מהצמתים שמקושרים לצומת זו, אם הוא עוד לא ביקר בהם.
צבעים: לבן –טרם התגלה, אפור- בטיפול, שחור – הסתיים הטיפול.

סיבוכיות: $O(|V|^3)$

דוגמת הרצה:



<u>משפטים וטענות מסביב:</u>

תכונת הסוגריים – אם $u \neq v$ אזי האינטרוולים $[d(v), f(v)], [d(u), f(u)]$ הינם זרים או שאחד מוכל באחר.

טענה – אם $u \to v$ (כלומר אם קיים מסלול בגרף מ-u ל-v) אזי $d(v) < f(u)$.

למת המסלול הלבן – אם $u_1 \to u_2 \to \cdots \to u_m$ ו- $d(u_1) < d(u_2), \dots, d(u_m)$ אזי $u_1 \overset{T}{\Rightarrow} u_m$

למת המסלול השחור – אם $u_1 \to u_2 \to \cdots \to u_m$ ו- $f(u_1), \dots, f(u_{m-1}) < f(u_m)$

מיון טופולוגי – יהא $G = (V, E)$ גרף מכוון. <u>מיון טופולוגי</u> של G הוא העתקה חד-חד ערכית $\varphi: V \to \mathbb{N}$ כך ש- $u \to v$ גורר $\varphi(u) < \varphi(v)$.

טענה – קיים מיון טופולוגי ל-*G* אם ורק אם *G* חסר מעגלים מכוונים.

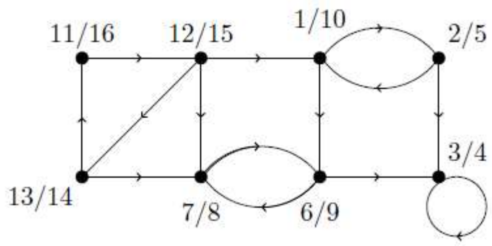
<u>קשירות חזקה באמצעות DFS</u>

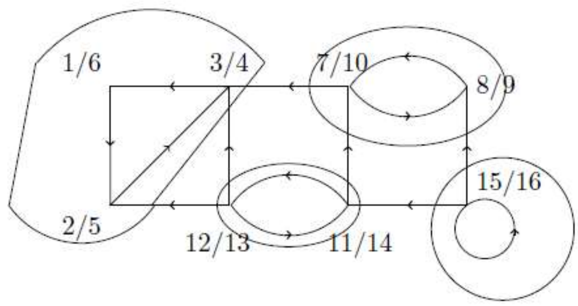
מטרה: מציאת מחלקות השקילות של יחס השקילות ~ בגרף מכוון.
(כאשר נגדיר $u \sim v$ אם $u \rightsquigarrow v$ ו- $v \rightsquigarrow u$)

תיאור האלגוריתם : הפעלת DFS על G וקבלת סדרת זמני הסגירה של הקודקודים ממנו. הפעלת DFS פעם נוספת, אך הפעם על G^t כאשר הקודקודים הנבחרים ב $DFS(G^t)$ נלקחים לפי סדר יורד של הסגירה מההפעלה הראשונה של DFS.

סיבוכיות: $O(|E|+|V|)$

דוגמת הרצה:





זרימה ברשתות

קצת מושגים...

- **רשת זרימה-** גרף מכוון עם זוג קודקודים מיוחדים: s נקרא מקור, t נקרא בור, ופונקציה אי שלילית $c : E \to \mathbb{R}^+$.
- c(e) נקרא הקיבול של הצלע e.
- $f : E \to \mathbb{R}^+$ תיקרא זרימה ברשת אם $0 \le f(e) \le c(e)$ לכל $e \in E$ ולכל קודקוד שונה מהמקור או מהבור מתקיים: $f^+(u) = f^-(u)$.
- ערך הזרימה מוגדר ע"י $val(f) = f^+(s) - f^-(s)$.
- לחתך s-t $(S,\bar{S})$ נגדיר את קבול החתך ע"י: $cap(S,\bar{S}) = \sum_{uv \in S\cdot\bar{S} \cap E} c(uv)$
- העודף של f על צלע $e_i \in P$ יוגדר ע"י:

$$\varepsilon_{f,P}(e_i) = \begin{cases} c(e_i) - f(e_i) & e_i = v_{i-1}v_i \\ f(e_i) & e_i = v_i v_{i-1} \end{cases}$$

- העודף של f על P יוגדר להיות המינימלי מבין העודפים על כל הצלעות ששייכות למסלול P.
- P מסלול s-t ייקרא f- לא רווי אם $\underline{\varepsilon}_{f,P} > 0$.

טענה- לכל זרימה f ולכל חתך s-t $(S,\bar{S})$ מתקיים: $val(f) = f(S,\bar{S}) - f(\bar{S},S)$.

דואליות חלשה- לכל זרימה f ולכל חתך s-t $(S,\bar{S})$ מתקיים: $val(f) \le cap(S,\bar{S})$.

טענה- אם P מסלול s-t אזי קיימת זרימה $\tilde{f}$ המקיימת $val(\tilde{f}) = val(f) + \underline{\varepsilon}_{f,P}$.

משפט הזרימה המקסימלית והחתך המינימלי – MFMC:

$$\max\{val(f) \ : \ \text{זרימה ברשת } f\} = \min\{cap(S,\bar{S}) \ : \ (s,t) \text{ חתך } (S,\bar{S})\}$$

טענה –

(i) אם $uv \in S \times \bar{S} \cap E$ אזי $f(uv) = c(uv)$

(ii) אם $vu \in \bar{S} \times S \cap E$ אזי $f(vu) = 0$

משפט – אם כל הקיבולים ברשת טבעיים אזי יש זרימה מקסימלית שלמה f.

טענה – קיים מסלול s-t – f-לא רווי ↔ קיים בG_f מסלול מכוון מ s ל-t.

אלגוריתם Edmonds-Karp

מטרה: מציאת זרימה אופטימלית ברשת זרימה.

תיאור האלגוריתם : מציאת מסלול השיפור p על ידי ביצוע אלגוריתם חיפוש לרוחב על הרשת, כל שהמסלול מ-s אל t יהיה מאורך מינימלי (אורך המסלול = מספר הקשתות שבו).

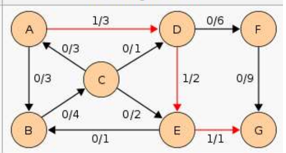
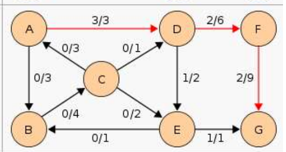
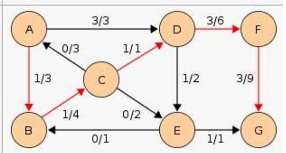
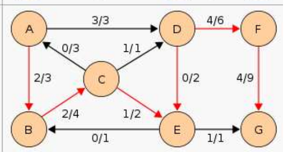
אלגוריתם:

1. מאתחלים את $f_1 = 0$
2. בהינתן f_k בנה את הגרף המכוון G_{f_k}.
   חפש מסלול מכוון מאורך מינימלי בין s לt על ידי BFS.
   אם אין מסלול כזה, סיים: $f = f_k$ היא זרימה מקסימלית ברשת.
   אחרת, יהא $g_1, \dots, g_m$ מסלול מינימלי בגרף מכוון ב- G_{f_k} מ-s ל-t.
   נגדיר זרימה חדשה f_{k+1} על G ע"י:

$$f_{k+1}(e) = \begin{cases} f_k(e) & e \notin \{e_1, \dots, e_m\} \\ f_k(e_i) + \varepsilon_{f_k,P} & g_i = \vec{e_i} \\ f_k(e_i) - \varepsilon_{f_k,P} & g_i = \overleftarrow{e}_i \end{cases}$$

סיבוכיות: $O(|E| \cdot |V|)$

דוגמת הרצה:

Capacity	Path	Resulting network
$\min(c_f(A,D), c_f(D,E), c_f(E,G))$ $= \min(3-0, 2-0, 1-0) =$ $= \min(3,2,1) = 1$	A, D, E, G	
$\min(c_f(A,D), c_f(D,F), c_f(F,G))$ $= \min(3-1, 6-0, 9-0)$ $= \min(2,6,9) = 2$	A, D, F, G	
$\min(c_f(A,B), c_f(B,C), c_f(C,D), c_f(D,F), c_f(F,G))$ $= \min(3-0, 4-0, 1-0, 6-2, 9-2)$ $= \min(3,4,1,4,7) = 1$	A, B, C, D, F, G	
$\min(c_f(A,B), c_f(B,C), c_f(C,E), c_f(E,D), c_f(D,F), c_f(F,G))$ $= \min(3-1, 4-1, 2-0, 0-(-1), 6-3, 9-3)$ $= \min(2,3,2,1,3,6) = 1$	A, B, C, E, D, F, G	

כיסויים וזיווגים בגרפים דו-צדדיים

קצת מושגים...

- $M \subset E$ נקראת **זיווג** אם $e_1 \cap e_2 = \emptyset$ לכל $e_1 \neq e_2 \in M$.
- $S \subset V$ נקרא **כיסוי** אם $S \cap e \neq \emptyset$ לכל $e \in E$.

$$\nu(G) = \max\{|M| \ : \ \text{זווג } M \subset E\}$$

$$\tau(G) = \min\{|S| \ : \ \text{כיסוי } S \subset V\}$$

משפט $\boldsymbol{König}$: אם G גרף דו צדדי אזי מתקיים $\nu(G) = \tau(G)$.

בהוכחה נעזרים בשתי הטענות הבאות:
(i) לכל זרימה שלמה f ב־H מתקיים $val(f) \leq \nu(G)$.
(ii) לכל חתך $s-t$ $(S,\bar{S})$ ב־H מתקיים $\tau(G) \leq Cap(S,\bar{S})$.

משפט $\boldsymbol{Hall}$: אם $|\Gamma(I)| \geq |I|$ לכל $I \subset A$ אזי G מכיל זווג המכסה את כל קודקודי A.

קשירות צלעית בגרפים מכוונים

קצת מושגים...

- מספר מקסימלי של מסלולים מכוונים זרים בקשתות מ־ s ל־ t $= \lambda_G(s,t)$
- מספר מינימלי של קשתות ב־E שהשמטתן מנתקת את s מ־t $= k_G(s,t)$

משפט $\boldsymbol{Menger}$: $\lambda_G(s,t) = k_G(s,t)$

בהוכחה נעזרים בשתי הטענות הבאות:

(i) לכל זרימה שלמה f ב־G מתקיים $val(f) \leq \lambda_G(s,t)$.
(ii) לכל חתך $s-t$ $(S,\bar{S})$ ב־G מתקיים $Cap(S,\bar{S}) \geq k_G(s,t)$.

צירקולציות

יהא $G=(V,E)$ גרף מכוון ופונקציות $\alpha,\beta : E \to \mathbb{R}$ כך שלכל $e \in E$

$$\alpha(e) \leq \beta(e)$$

צירקולציה ב־G היא העתקה $g : E \to \mathbb{R}$ המקיימת

(i) $g^+(u) = g^-(u)$ לכל $u \in V$

(ii) $\alpha(e) \leq g(e) \leq \beta(e)$

נגדיר רשת חדשה $H = (V', E')$ $\quad V' = \{s,t\} \cup V$

$$E' = E \cup \{su \ : \ u \in V\} \cup \{ut \ : \ u \in V\}$$

עם קבולים

$$c(e) = \begin{cases} \beta(e) - \alpha(e) & e = uv \in E \\ \alpha(V,u) & e = su \\ \alpha(u,V) & e = ut \end{cases}$$

כאשר כמקודם $\alpha(V,u) = \sum_{v \in V} \alpha(vu) \quad , \quad \alpha(u,V) = \sum_{v \in V} \alpha(uv)$

טענה- קיימת צירקולציה ב-G אם ורק אם קיימת זרימה ב-H שערכה $\alpha(V,V)$.

משפט $\boldsymbol{Hoffman}$ – קיימת צירקולציה ב-G אם ורק אם לכל חתך $(S,\bar{S})$ מתקיים: $\alpha(S,\bar{S}) \leq \beta(\bar{S},S)$

אלגוריתמים אריתמטיים והצפנה צבורית:

אלגוריתם אוקלידס למציאת gcd:

מטרה: מציאת הגורם המשותף הגדול ביותר בין שני מספרים.

אלגוריתם: יהיו $b \geq a \geq 0$ ‏ $(a,b) \neq (0,0)$.
נגדיר $c_1 = b$, $c_2 = a$. נניח שהגדרנו $c_1, \ldots, c_k$ ‏ $k \geq 2$.
אם $c_k = 0$ אזי $gcd(a,b) = c_{k-1}$. אחרת נחלק עם שארית $c_{k-1} = 0 \leq a_{k+1} < a_k$
‏$q_k c_k + c_{k+1}$, ונקבל על ידי זה את c_{k+1}.

סיבוכיות: $O\left(\max\{\log_\varphi a, \log_\varphi b\}\right) = \max\{\frac{\log_2 a}{\log_2 \varphi}, \frac{\log_2 b}{\log_2 \varphi}\}$ כאשר φ הוא מספר הזהב: $\frac{1+\sqrt{5}}{2}$

דוגמת הרצה: $gcd(57,72) = gcd(15,57) = gcd(12,15) = gcd(3,12) = gcd(0,3) = 3$

<u>טענה-</u> לכל $(a.b) \in \mathbb{Z}^2$ ‏ $(0,0) \neq$ קיימים $x, y \in \mathbb{Z}$ כך ש- $\gcd(a,b) = x \cdot a + y \cdot b$.

דוגמא לשימוש באלגוריתם אוקלידס ובטענה:

למשל: יהיו $a = 2322,\ b = 654$.

$2322 = 654 \cdot 3 + 360$	$(2322, 654) = (654, 360)$
$654 = 360 \cdot 1 + 294$	$(654, 360) = (360, 294)$
$360 = 294 \cdot 1 + 66$	$(360, 294) = (294, 66)$
$294 = 66 \cdot 4 + 30$	$(294, 66) = (66, 30)$
$66 = 30 \cdot 2 + 6$	$(66, 30) = (30, 6)$
$30 = 6 \cdot 5$	$(30, 6) = 6$

כדי להציג את המ.מ.מ. בצורה $\alpha a + \beta b$, נרשום $6 = 66 - 30 \cdot 2$ (מהשוויון הלפני אחרון), במקום ה 30 נציב את ערכו מהמשוואה שלפניה, ואז את 66 מהמשוואה הקודמת לזו, כך נמשיך ונקבל: $2322 \cdot 20 + 654 \cdot (-71) = 6$.

<u>משפט השאריות הסיני</u> – יהיו $n_1, \ldots, n_k$ זרים בזוגות אזי F היא איזומורפיזם של חבורות אבליות
$F: \mathbb{Z}^*_{n_1,\ldots,n_k} \to \mathbb{Z}^*_{n_1} \times \ldots \times \mathbb{Z}^*_{n_k}$.

<u>דוגמא:</u> $(n_1, n_2, n_3) = (5,6,7)$

$$\alpha_1 = (n_2 n_3)^{-1}(mod\ n_1) = 42^{-1}(mod\ 5) = 3$$
$$\alpha_2 = (n_1 n_3)^{-1}(mod\ n_2) = 35^{-1}(mod\ 6) = 5$$
$$\alpha_3 = (n_1 n_2)^{-1}(mod\ n_3) = 30^{-1}(mod\ 7) = 4$$

לכן

$$\begin{aligned} x &= 42 \cdot 3a_1 + 35 \cdot 5a_2 + 30 \cdot 4a_3 \\ &= 126a_1 + 175a_2 + 120a_3 \end{aligned}$$

מקיים $x(mod\ n_i) = a_i$ לכל $1 \leq i \leq 3$.

טענה- אם $n_1, \ldots, n_k$ זרים בזוגות אזי ההעתקה $F: \mathbb{Z}_{n_1,\ldots,n_k} \to \mathbb{Z}_{n_1} \oplus \ldots \oplus \mathbb{Z}_{n_k}$ הנתונה ע"י $F(x) = (x(mod\ n_1), \ldots, x(mod\ n_k))$ היא איזומורפיזם של חוגים.

טענה- אם $n = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ כאשר p_i ראשוניים ו - $\alpha_i \geq 1$ אזי :

$$\varphi(n) = \sum_{i=1}^{k}(p_i^{\alpha_i} - p_i^{\alpha_{i-1}}) = n \cdot \prod_{i=1}^{k}(1 - \frac{1}{p_i})$$

המשפט הקטן של פרמה: אם $\gcd(a,n) = 1$ אזי $a^{\varphi(n)} \equiv 1(mod\ n)$.

סיבוכיות של פעולות אריתמטיות ב- $\mathbb{Z}_n$:

- חיבור $O(\log n)$
- חיסור $O(\log^2 n)$
- חילוק עם שארית $O(\log^2 n)$
- העלה בחזקה $O(\log^3 n)$
- כפל $O(\log^2 n)$
- חישוב הופכי $O(\log^2 n)$

בעיות הכרעה

קצת מושגים...

- $\{0,1\}^*$ כל הסדרות הסופיות של 0,1
- $L \subset \{0,1\}^*$ תת קבוצה הנקראת <u>שפה</u>.
- <u>בעיית הכרעה-</u> בהינתן $x \in \{0,1\}^*$ קבע האם $x \in L$ או לא.
- <u>אלגוריתם דטרמיניסטי –</u> לשייכות ל-L הוא אלגוריתם A המחשב לכל $x \in \{0,1\}^*$: $A(x) \in \{0,1\}^*$.
  $L = \{x \in \{0,1\}^* : A(x) = 1\}$.
- A ייקרא <u>פולינומיאלי</u> אם זמן הריצה של A על x הוא $O(|x^c|)$ עבור c קבוע שלא תלוי ב-x.
- <u>אלגוריתם טוב</u> הוא אלגוריתם פולינומיאלי.
- אלגוריתם הסתברותי ε-טוב לשפה L הוא אלגוריתם B המקבל כקלט $x \in \{0,1\}^*$ ומשתנה מקרי ω (המייצג מספר הטלות מטבע) כך ש:
  - אם הפלט של B הוא 0 אזי בביטחון $x \notin L$.
  - אם $x \notin L$ יש הסתברות קטנה מ ε ש-B יחליט אחרת, ש- $x \in L$.
  - אם $x \notin L$ אז בהסתברות גדולה מ $1-\varepsilon$ ש-B יחליט כי $x \notin L$.

מציאת ראשוניים גדולים

משפט המספרים הראשוניים PNT – PNT אומר שאם נעיין במספרים $x, x+1, \dots, x+10\ln x$ אז בדרך כלל יהיה ביניהם מספר ראשוני. $\pi(\text{x}) = \left|\{p : p \le x \text{ ראשוני } \}\right|$, אזי $\pi(\text{x}) \sim \frac{x}{\ln x}$ במובן ש- $\lim_{x\to\infty} \frac{\pi(\text{x})}{\frac{x}{\ln x}} = 1$.

עדים לפריקות של n –

יהא $n > 1$ מספר אי-זוגי, ונכתוב $n - 1 = 2^t u$ כאשר $2 \nmid u$.

$$W_1(n) = \{a \in \mathbb{Z}_n^* \ : \ a^{n-1} \not\equiv 1 (mod\ n)\}$$

<u>אם $\text{W}_1(n) \neq \phi$ אז n פריק!</u>

מספר פריק n עבורו $\text{W}_1(n) = \phi$ נקרא <u>מספר *Caramichael*</u>, למשל 561. ידוע שיש אינסוף מספרים כאלו.

$$W_2(n) = \{a \in \mathbb{Z}_n^* \ : \ \exists 1 \le i \le t \ a^{2^{i-1}u} \not\equiv \pm 1 (mod\ n) \ , \ a^{2^i u} \equiv 1 (mod\ n)\}$$

<u>אם $\text{W}_2(n) \neq \phi$ אז n פריק!</u>

האלגוריתם ההסתברותי של מילר-רבין לבדיקת ראשוניות

מטרה: בדיקת ראשוניות של מספר.

אלגוריתם: נציג $n - 1 = 2^t u$ כאשר $2 \nmid u$. יהא $s = \lceil \log_2 \frac{1}{\varepsilon} \rceil$.
עבור $1 \le j \le s$ נבצע את התהליך הבא: נגריל $1 < a_j < n$ מקרי.

I. אם $\gcd(a_j, n) > 1$ הכרז n <u>פריק</u>, סיים.
אחרת, חשב את הסדרה $a_j^u, a_j^{2u}, \dots, a_j^{2^t u} = a_j^{n-1}$.

II. אם $a_j^{n-1} \not\equiv \pm 1(n)$ הכרז n <u>פריק</u>, סיים.

III. אחרת, יהא i המינימלי כך ש- $a_j^{2^i u} \equiv 1(n)$.
אם $a_j^{2^{i-1}u} \not\equiv \pm 1(n)$ הכרז n <u>פריק</u>, סיים.

IV. אם בכל s האיטרציות n לא הוכרז פריק, הכרז n <u>ראשוני</u>.

סיבוכיות: האלגוריתם הוא אלגוריתם הסתברותי ε-טוב מכיוון שאם הכרזנו שהמספר פריק, הוא באמת פריק. אם הכרזנו שהמספר ראשוני בשלב IV, יכול להיות שההכרזה לא נכונה, במקרה כזה הטעות קטנה מ-ε . הסיבוכיות היא $O(\log\frac{1}{\varepsilon} \cdot \log^3 n)$.

הצפנה ציבורית בשיטת RSA-

<u>שיטת ההצפנה הקלאסית</u> – מניחים כי כל הודעה מורכבת מ-n אותיות לועזיות ושכל אות מיוצגת ע"י j, כאשר $0 \leq j \leq 25$. מחליטים מראש על פונקציית ההצפנה הסודית E ופונקציית פענוח סודית $D = E^{-1}$.

הבעיות שעולות מההצפנה הקלאסית- קשה לייצר סדרות שדומות למקריות, השיטה מחייבת תיאום מראש.

<u>שיטת ההצפנה הציבורית –</u>

על מנת לקבל מרחב הודעות גדול ממספר משתמשים, ללא פגישה איתם מראש, עלינו לבחור פונקציה חד-חד ערכית ועל $E: M \to \mathcal{M}$. על ההצפנה הציבורית לקיים מספר תנאים (פונקציה כזו נקראת trapdoor function):

1. E ניתנת לחישוב מהיר על כל קלט $M \in \mathcal{M}$.
2. אנו יכולים לחשב את D(פונקציית הפענוח) במהירות על כל $M \in \mathcal{M}$
3. למרות ש-E נתונה לכל, איש מלבידנו לא יכול לחשב את D ($= E^{-1}$) בזמן סביר.

<u>שיטת RSA –</u>

- בוחרים שני מספרים ראשוניים p, q גדולים.
- מחשבים $n = pq$. מרחב ההודעות מוגדר להיות $\mathcal{M} = \mathbb{Z}_n^*$.
- בוחרים מספר גדול e שמקיים $gcd\big(e, (p-1)(q-1)\big) = 1$.
- מפרסמים את הזוג (n, e) לכל.
- מגדירים את פונקציית ההצפנה $E: \mathcal{M} \to M$ ע"י: $\forall x \in \mathcal{M} : E(x) = x^e (mod\ n)$
- בעזרת אלגוריתם אוקלידס, אנו מחשבים את $d = e^{-1}$ כך ש- $de \equiv_{(p-1)(q-1)} 1$
- הפונקציה ההופכית ל-*E* נתונה ע"י: $\forall y \in \mathcal{M} : D(y) \equiv y^d (mod\ n)$.
- נוודא שהפונקציה *D* אכן הופכית ל-*E*:

$$\forall x \in \mathcal{M}: D\big(E(x)\big) = (x^e)^d mod\ n = x^{1+\lambda(p-1)(q-1)} mod\ n = x^{1+\phi(n)} (mod\ n) \equiv_n x$$

דוגמא להרצה-

נתון מפתח ציבורי *n=77* ו*e=13*. התקבלה הודעה 36, פענחו את ההודעה.

<u>פיתרון-</u> מכיוון שאנו יודעים לפרק את 77 לגורמים, נוכל למצוא את ההודעה. 77=7*11.

ולכן צריך למצוא הופכי ל-13 ב- $\mathbb{Z}_{60}^*$ (כי $x^{13} =_{77} 36$). יש כזה כי 13 ו-60 זרים.

ע"י אלגוריתם אוקלידס:

$(60{,}13),\ 60 = 4 \cdot 13 + 8$	$(13{,}8),\ 13 = 1 \cdot 8 + 5$	$(8{,}5),\ 8 = 1 \cdot 5 + 3$
$(5{,}3),\quad 5 = 1 \cdot 3 + 2$	$(3{,}2),\ 3 = 1 \cdot 2 + 1$	$(2{,}1),\ 2 = 1 \cdot 2 + 0$
$(1{,}0),\quad 1 = gcd$		

נרצה לחלץ את 1 - $1 = 5 \cdot 60 + 23 \cdot 13$, $d \equiv_{60} -23 \equiv_{60} 37$.

כעת עלינו לחשב את $(36)^{37}\ (mod\ 77) = 64$.

<u>בעיות אופטימיזציה ובעיות הכרעה-</u>

בעיות אופטימיזציה – עץ פורש מינימלי, המסלול הקצר ביותר בין זוג קודקודים בגרף, מציאת זרימה מקסימלית ברשת, בעיות הכרעה- בעיות שהתשובה עליהן היא כן או לא.

"טענה" – נניח p בעיית אופטימיזציה, q בעיית ההכרעה הנגזרת ממנה. אזי אם ל-q יש אלגוריתם פולינומיאלי, גם ל-p יש.

טענה – אם יש אלגוריתם שמכריע את q בזמן פולינומיאלי אז יש אלגוריתם A שפותר את p בזמן פולינומיאלי.

<u>המחלקה *P* – אוסף השפות הפולינומיאליות</u>

דוגמאות לשפות ב-*P*:

- שפת הגרפים הדו-צדדיים.
- שפת הגרפים הדו-צדדיים בעלי זיווג מושלם.
- שפת הגרפים הממושקלים בעלי עץ פורש ≥ 10.
- שפת הגרפים המכילים קליק (גרף שלם) בגודל ≤ 10.

<u>המחלקה *NP*</u>

שפות רבות L שעבורן אם $x \in L$ אזי אורקל (כוח עליון) יכול להוכיח לנו בזמן פולינומיאלי שאכן $x \in L$. בשפה פורמלית,

$L \in NP$ אם קיים אלגוריתם $A : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ וקבוע $c > 0$ כך ש־ $A(x,y)$ מחושב בזמן $O((|x|+|y|)^c)$ וכך ש־

$$L = \{x \in \{0,1\}^* \ : \ \exists y \in \{0,1\}^* \ , \ |y| \leq O(n^c) \ , \ A(x,y) = 1\}$$

דוגמא לשפה ב-*NP*:

- מעגל המילטוני הוא מעגל פשוט העובר דרך כל קודקודי G. אם $G \in HAM$ אזי אורקל יכול לספק לנו סידור של הקודקודים ואנו נוכל לבדוק בזמן לינארי ב- n שהסידור מגדיר מעגל המילטוני.

טענה - $P \subset NP$ (נתעלם מאורקל – כלומר גם בלי האורקל נוכל לקבוע אם $x \in L$ או $x \notin L$.

הגדרה - השפה L_1 ניתנת לרדוקציה פולינומיאלית לשפה L_2, ובסימון $L_1 \leq_p L_2$, אם קיים אלגוריתם

$f: \{0,1\}^* \to \{0,1\}^*$ וקיים קבוע $c > 0$ כך שמתקיימים התנאים הבאים:

1. $f(x)$ מחושב בזמן $O(|x|^c)$.
2. $x \in L_1 \Leftrightarrow f(x) \in L_2$.

טענה – לכל k קבוע, $COL(k) \leq_P COL(k+1)$

טענה - $SAT \leq_P 3CNF$

טענה –

1. אם $L_2 \leq_p L_3$ וגם $L_1 \leq_p L_2$ אזי $L_1 \leq_p L_3$
2. אם $L_1 \leq_p L_2$ ו- $L_2 \in P$ אזי $L_1 \in P$

המחלקה *NPC*

השפה *L* נקראת <u>*NP*-קשה</u> אם $L' \leq_p L$ לכל $L' \in NP$.

L נקראת <u>*NP*-שלמה</u> (*NPC*) אם $L \in NP$ ו-*L* היא *NP*- קשה.

<u>משפט קוק -</u> $SAT \in NPC$ או $CSAT \in NPC$.

SAT - נוסחה בוליאנית $\phi(x_1, \dots, x_n)$ נקראת <u>ספיקה</u> אם יש הצבה $x = (x_1, \dots, x_n) \in (T, F)^n$ כך ש- $\phi(x) = T$. ($\mathrm{T} - \mathrm{true}, \mathrm{F} - \mathrm{false}$).

דוגמאות לשפות ב-*NPC*:

- נוסחה בוליאנית ϕ היא ב-CNF אם ϕ היא AND של OR-ים באורך ≤ שלוש של משתנים או שלילותיהם. נסמן ב-**3CNF** את אוסף הנוסחאות הנ"ל שהן ספיקות.
- נסמן ב $\omega(G)$ את גודל הקליק המרבי בגרף G.
  נסמן ב-CLIQUE את אוסף הזוגות (G, k) כך ש- $\omega(G) \geq k$.
- נסמן ב-COVER את אוסף הזוגות (G, k) כך ש- $\tau(G) \leq k$.
- נסמן ב-DHAM את אוסף הגרפים המכוונים G בעלי מעגל המילטוני מכוון.

מתקיים כי - $3CNF, CLIQUE, COVER, HAM, DGAM, COL(3) \in NP$.

טענה – $CSAT \leq_p SAT \leq_p 3CNF \leq_p CLIQUE \leq_p COVER \leq_p \mathrm{DHAM}$

$3CNF \leq_p \mathrm{COL}(3)$. ניתנה בהרצאה הוכחה לכל אחד מסימני האי-שוויון.

טענה - $\tau(\bar{G}) + \omega(G) = |V|$