זרימה ברשתות

רשת זרימה (flow network) $N(G(V,E),s,t,c)$ מורכבת מחלקים הבאים:

- $G(V,E)$ – גרף מכוון, סופי ופשוט
- s – צומת **מקור** (source)
- t – צומת **בור** (sink, target)
- c – פונקציית **קיבול** (capacity function) $c: E \to \mathbb{R}^+$ המגדירה לכל קשת את כמות הזרימה המירבית שניתן להזרים בקשת זו.

נסמן ב-$d_{in}(v)$ את דרגת הכניסה (input degree) של צומת v, וב-$d_{out}(v)$ את דרגת היציאה (output degree) שלו.
נסמן ב-$in(v)$ את קבוצת הקשתות שנכנסות לצומת v, וב-$out(v)$ את קבוצת הקשתות שיוצאות ממנו.
זרימה ברשת היא פונקציה $f: E \to \mathbb{R}^+$ המקיימת:

- חוק הקשת: זרימה בכל קשת היא אי שלילית ואינה חורגת מקיבולה של הקשת
$$\forall e \in E \quad 0 \le f(e) \le c(e)$$
- חוק הצומת: זרימה לא נוצרת ולא נצברת באף צומת חוץ ממקור ובור (אולי).
$$\forall v \in V \setminus \{s,t\} \qquad \sum_{e \in in(v)} f(e) - \sum_{e \in out(v)} f(e) = 0$$

עוצמת הזרימה *(או כמות הזרימה, או ערך הזרימה)* F מוגדרת כסה"כ הזרימה שנצברת בבור $F = \sum_{e\in in(t)} f(e) - \sum_{e \in out(t)} f(e)$.
זרימת מקסימום (maximum flow) היא פונקציית הזרימה שממקסמת את עוצמת הזרימה ברשת (כלומר, זאת פונקציה שמזרימה כמות זרימה מקסימלית אפשרית בתוך הרשת).
קשת רוויה (saturated edge) היא קשת שהזרימה בה שווה לקיבולה. כלומר, אם קשת e רוויה, אזי מתקיים $f(e) = c(e)$.
זרימה רוויה זאת פונקצית זרימה עבורה מתקיים שבכל מסלול מכוון מ-s ל-t קיימת לפחות קשת רוויה אחת.
חתך s-t (s-t cut) $(S,\bar{S})$ ברשת $N(G(V,E),s,t,c)$ הוא חלוקה של צמתי הרשת לשתי קבוצות S ו-$\bar{S} = V \setminus S$ כך ש-$s \in S$ ו-$t \in \bar{S}$.
קשת $e = (u,v)$ **שייכת לחתך** $(S,\bar{S})$, אם $u \in S$ ו-$v \in \bar{S}$. סימון: $e \in (S,\bar{S})$ או $(u,v) \in (S,\bar{S})$.
קשת (u,v) **שייכת אחורנית לחתך** $(S,\bar{S})$ *(חוזרת בחתך, הפוכה בחתך)*, אם $v \in S$ ו-$u \in \bar{S}$. נסמן: $(u,v) \in (\bar{S},S)$.
זרימה בחתך $(S,\bar{S})$ היא סה"כ כמות הזרימה שעוברת מ-S ל-$\bar{S}$. סימון: $f(S,\bar{S})$.
חישוב: $f(S,\bar{S}) = \sum_{e\in(S,\bar{S})} f(e) - \sum_{e\in(\bar{S},S)} f(e)$.

למת הזרימה בחתך

לכל חתך s-t $(S,\bar{S})$ מתקיים שהזרימה בו שווה לעוצמת הזרימה ברשת, כלומר $f(S,\bar{S}) = F$.
קיבול החתך $(S,\bar{S})$ (cut capacity) (מסומן $c(S,\bar{S})$) מוגדר כסכום קיבולי הקשתות ששייכות לחתך: $c(S,\bar{S}) = \sum_{e\in(S,\bar{S})} c(e)$.
חתך s-t מינימום (minimum cut) הינו חתך בעל קיבול מינימלי בין כל חתכי s-t ברשת. (יתכנו כמה חתכי מינימום באותה רשת)

למת קיבול החתך

קיבול של כל חתך s-t $(S,\bar{S})$ חוסם מלמעלה את עוצמת הזרימה ברשת, כלומר $c(S,\bar{S}) \ge F$.
קשת e **מועילה קדימה**, אם אינה רוויה. כלומר $c(e) > f(e)$.
קשת e **מועילה אחורה**, אם אינה ריקה. כלומר $f(e) > 0$.
בהנתן רשת זרימה $N(G(V,E),s,t,c)$ וזרימה f בה, ניתן להגדיר **רשת שיורית** $N_f(G_f(V,E_f),s,t,c_f)$ באופן הבא:

- לכל קשת $(u,v) \in E$ שמועילה קדימה מוגדרת **קשת שיורית קידמית** $(u,v) \in E_f$ עם **קיבול שיורי** $c_f(u,v) = c(u,v) - f(u,v)$.
- לכל קשת $(u,v) \in E$ שמועילה אחורה מוגדרת **קשת שיורית אחורית** $(v,u) \in E_f$ עם **קיבול שיורי** $c_f(v,u) = f(u,v)$.

בהנתן רשת זרימה $N(G(V,E),s,t,c)$ וזרימה f בה, **מסלול שיפור** הוא מסלול מכוון מ-s ל-t המורכב מקשתות שיוריות. נוח לחפש מסלולי שיפור ברשת שיורית.

למת מסלול שיפור

תהי f זרימה חוקית ברשת $N(G(V,E),s,t,c)$ ו-F עוצמת הזרימה, יהי P מסלול שיפור, ויהי B הקיבול השיורי המינימלי לאורך המסלול. אם נגדיל ב-B את הזרימות בכל הקשתות המתאימות לקשתות הקידמיות של P, ונקטין ב-B את הזרימות בכל הקשתות המתאימות לקשתות האחוריות של P, נקבל זרימה חוקית חדשה שעוצמתה היא $F + B$.

משפט חתך מינימום – זרימת מקסימום (Min Cut – Max Flow)

תהי f זרימה חוקית ברשת $N(G(V,E),s,t,c)$ הטענות הבאות הינן שקולות:

1. *f היא זרימת מקסימום.*
2. *לא קיים מסלול שיפור מ-s ל-t.*
3. *קיים ברשת חתך s-t $(S,\bar{S})$ אשר קיבולו שווה לעוצמת הזרימה ברשת (חתך מינימום).*

אלגוריתם Ford-Fulkerson

1. ברשת נתונה $N(G(V,E),s,t,c)$ **הגדר** זרימה התחלתית חוקית f (למשל, $\forall e \in E \quad f(e) = 0$).
2. **חפש** מסלול שיפור P מ-s ל-t. (ניתן להעזר ברשת שיורית, אך לא הכרחי)
3. **אם** לא נמצא מסלול שיפור – **סיים** (הזרימה הקיימת היא מקסימום).
4. **אחרת**, **מצא** קיבול שיורי מינימלי לאורך המסלול (נסמנו B).
5. **שפר** את הזרימה ברשת N בעזרת המסלול P באופן הבא:
   a. לכל קשת קידמית $(u,v) \in P$ **בצע** לקשת המקורית המתאימה $(u,v) \in E$ עדכון $f(u,v) \leftarrow f(u,v) + B$
   b. לכל קשת אחורית $(u,v) \in P$ **בצע** לקשת המקורית המתאימה $(v,u) \in E$ עדכון $f(v,u) \leftarrow f(v,u) - B$
6. חזור ל-2

ביצוע פעולות 2-5 פעם אחת מהווה ***פאזה*** אחת של האלגוריתם.

אלגוריתם Edmonds-Karp

הרץ אלגוריתם FF כאשר בכל פאזה יש לבחור מסלול שיפור הקצר ביותר מבין כל מסלולי שיפור קיימים.
כלומר, מוצאים מסלול שיפור בעזרת הרצת BFS מצומת s ברשת שיורית.
סיבוכיות: $O(|V| \cdot |E|^2)$.

אלגוריתם Dinitz

הרץ אלגוריתם FF כאשר בכל פאזה יש לבחור את כל מסלולי השיפור הקצרים ביותר שיש ברשת.
בתחילת כל פאזה בונים רשת שכבות בה כל שכבה מכילה צמתים הנמצאים באותו מרחק מ-s ברשת שיורית מתאימה. בעזרת רשת השכבות מאתרים את כל מסלולי השיפור הקצרים ביותר מ-s ל-t ומנצלים אותם.
הסיבוכיות של האלגוריתם היא $O(|E| \cdot |V|^2)$.

זרימה בשלמים היא פונקציה המזרימה מספר שלם על כל קשת.

רשתות 0-1

רשת זרימה 0-1 הינה רשת זרימה בה כל הקיבולים הם 1.
זרימה 0-1 היא פונקציה המזרימה בכל קשת רק 0 או 1.
ברשת זרימה עם קיבולים שלמים סיבוכיות האלגוריתם של Ford ו-Fulkerson היא $O(|E| \cdot M)$ כאשר M הוא ערך זרימת מקסימום.
ברשת 0-1 סיבוכיות האלגוריתם של Ford ו-Fulkerson היא $O(|E| \cdot |V|)$.
גם לאלגוריתם של Dinitz, סיבוכיות $O(|E| \cdot |V|)$.

שידוכים בגרפים

שידוך *(זיווג)* (matching) בגרף לא מוונך $G(V,E)$ הינו קבוצת קשתות $M \subseteq E$ כך שלכל זוג קשתות בה אין קצה משותף. כלומר, לכל זוג קשתות $(u,v),(x,y) \in M$ הצמתים u,v,x,y הם בהכרח שונים.
צומת $v \in V$ ***משודך*** *(מכוסה)* (covered) ע"י שידוך M, אם קיימת קשת $(v,u) \in M$.
צומת $v \in V$ ***חשוף*** (exposed) יחסית לשידוך M, אם אינו משודך ב-M.
שידוך מקסימום הינו שידוך גדול ביותר האפשרי בגרף הנתון. (גודל שידוך נמדד במספר הקשתות בו)
שידוך רווי *(מקסימלי)* הינו שידוך שלא ניתן להרחבה ע"י הוספת קשתות אליו.
שידוך מושלם (perfect matching) הינו שידוך בו כל צומת $v \in V$ משודך.

מציאת שידוך רווי

בהנתן גרף לא מכוון $G(V,E)$ ניתן למצוא בו שידוך רווי ע"י אלגוריתם פשוט הבא:

- $M \leftarrow \emptyset$
- כל עוד $E \neq \emptyset$ בצע
  a. בחר שרירותית קשת $(u,v) \in E$
  b. בצע $M \leftarrow M \cup \{(u,v)\}$
  c. מחק מ-E את כל הקשתות המחוברות ל-u או ל-v

סיבוכיות: האלג' רץ בזמן $O(|E|)$.

גרף דו-צדדי (bipartite graph, bigraph) גרף (לא מוונך בד"כ) $G(V,E)$ בו $V = L \cup R$ כאשר $L \cap R = \emptyset$ ו-$E \subseteq L \times R$.
לפעמים, רושמים את הגרף $G(L \cup R, E)$.
אם $|L| = |R|$, אז הגרף הדו-צדדי נקרא ***מאוזן***.
רשת עזר:
בהנתן גרף דו-צדדי $G(L \cup R, E)$ נבנה רשת זרימה $N(G'(V',E'),s,t,c)$ באופן הבא:

- $V' = L \cup R \cup \{s,t\}$ (כאשר s ו-t הם צמתים חדשים)
- $E' = \{s \to v | v \in L\} \cup \{u \to v | u \in L, v \in R, (u,v) \in E\} \cup \{v \to t | v \in R\}$
- $\forall e \in E' \quad c(e) = 1$ (כלומר, זאת רשת 0-1)

למת זרימה ושידוך

ב-N קיימת זרימה בשלמים בעוצמה k אמ"מ ב-G קיים שידוך בגודל k.

אלגוריתם למציאת שידוך מקסימום בגרף דו-צדדי

1. בהנתן גרף דו-צדדי $G(L \cup R, E)$ **בנה** רשת עזר $N(G'(V',E'),s,t,c)$ כמתואר לעיל.
2. **מצא** זרימת מקסימום ב-N.
3. **בחר** לשידוך את כל הקשתות המקוריות בהן הזרימה שונה מ-0.

סיבוכיות: $O(|E| \cdot |V|)$ (ע"י FF או Dinitz, למשל). חסם יותר הדוק הוא $O\left(|E| \cdot \min\{|L|,|R|\}\right)$.

בהנתן גרף דו-צדדי $G(L \cup R, E)$ וקבוצת צמתים $A \subseteq L$, ***קבוצת שכנים*** של A – $\Gamma(A)$ (סימון מקובל אחר $N(A)$) – מוגדרת כקבוצת כל הצמתים ב-R שיש להם שכנים ב-A.
כלומר, $\Gamma(A) = \{v \in R | (u,v) \in E, u \in A\}$.

משפט Hall

בגרף דו-צדדי $G(L \cup R, E)$ בו $|L| = |R|$ קיים שידוך מושלם אמ"מ לכל $A \subseteq L$ מתקיים $|\Gamma(A)| \geq |A|$. (באופן כללי יותר: עבור $|L| \leq |R|$ קיים שידוך המכסה את כל צמתי L.)

הבעיה Subset Sum

נתונה קבוצה $\{a_1,a_2,....,a_n\}$ של מספרים טבעיים ומספר מטרה S. האם קיימת תת-קבוצה שסכומה S .

<u>פתרון פסאודו-פולינומי:</u> נגדיר פונקציה **בולאנית** ונציג לה פתרון מתמטי רקורסיבי:

Sum(i,t) – האם קיימת תת קבוצה של $\{a_1,a_2,....,a_i\}$ שסכומה t

<u>תנאי התחלה:</u>

$$Sum(0,t) = FALSE \qquad \forall\ 1 \le t \le S$$
$$Sum(i,0) = TRUE \qquad \forall\ 0 \le i \le n$$
$$Sum(i,t) = FALSE \qquad \forall\ t<0\ ,\quad \forall\ 0 \le i \le n$$

<u>מקרה כללי:</u>

$$Sum(i,t) = Sum(i-1,t)\ \textbf{or}\ Sum(i-1,t-a_i) \qquad \forall\ 1 \le i \le n\ ,\ \forall\ 1 \le t \le S$$

והפתרון הוא: Sum(n,S).

<u>סיבוכיות</u> O(nS).

תורת הסיבוכיות

מכונת טיורינג

מכונת טיורינג מוגדרת ע"י שביעייה $M = (Q, q_0, F, \Sigma, \Gamma, \sqcup, \delta)$ אשר פרושה:

- Q – קבוצה <u>סופית</u> של *מצבים*.
- $q_0 \in Q$ – מצב התחלתי.
- $F \subseteq Q$ – קבוצת מצבים סופיים.
- Σ – א"ב הקלט (אוסף כל התווים העשויים להופיע בקלט).
- $\Gamma \supset \Sigma$ – א"ב עבודה (אוסף כל התווים הנמצאים בשימוש המכונה).
- $\sqcup \in \Gamma \setminus \Sigma$ – סימן *רווח*.
- δ – פונקציית מעברים $\delta: ((Q \setminus F) \times \Gamma) \to (Q \times \Gamma \times \{L,R,S\})$, כאשר $\{L,R,S\}$ מסמנים כיוון התנועה של *ראש קורא*.

<u>איתחול המכונה:</u>

הקלט $x \in \Sigma^*$ רשום על הסרט החל מהקצה השמאלי. מימין לקלט נמצא רצף אינסופי של רווחים $\sqcup$. הראש מצביע על התא השמאלי ביותר. הבקר נמצא במצב התחלתי q_0.

<u>פעולת המכונה:</u>

- בכל צעד החישוב המכונה מבצעת פעולות לפי פונקציית המעברים δ כתלות במצב הבקר הנוכחי ותוכן התא עליו מצביע הראש.
- אם הראש מצביע על התא השמאלי ביותר ויש להזיז שמאלה (לפי δ) אזי הראש נשאר במקום (לאו דווקא מסיים את הריצה).
- החישוב מסתיים כאשר המכונה נכנסת למצב סופי (אחד המצבים מ-F) – המכונה עוצרת והפלט שלה הוא כל מה שנמצא על הסרט החל מהקצה השמאלי ועד למיקום הראש (לא כולל).
- לפעמים מכונה לא עוצרת על קלט מסוים.

בעית הכרעה היא בעיה שבהינתן קלט מסוים הפלט הצפוי הוא "כן" או "לא".

מכונת טיורינג לזיהוי שפה היא מכונת טיורינג עם שני מצבים סופיים : $F = \{q_Y, q_N\}$ (q_Y מחזיר "כן", q_N מחזיר "לא").

זמן (בהקשר של מכונות טיורינג) הוא מספר צעדי החישוב של מכונת טיורינג.

סיבוכיות זמן היא פונקציה (אולי חלקית) $t_M : \Sigma^* \to \mathbb{N}$ (M – מכונת טיורינג) המוגדרת באופן הבא:

$$t_M(x) = \begin{cases} \text{x על M מספר צעדי חישוב של} & \text{אם M עוצרת} \\ \text{לא מוגדר} & \text{אחרת} \end{cases}$$

חסם סיבוכיות הוא פונקציה מלאה $T_M : \mathbb{N} \to \mathbb{N}$ כזאת שלכל $x \in \Sigma^*$ $T_M(|x|) \ge t_M(x)$.

מכונת טיורינג הינה ***פולינומית*** אם קיים פולינום $p(n)$ אשר מהווה חסם סיבוכיות עבורה.

תזת Church המורחבת (Edmonds)

בכל מודל כללי וסביר של חישוב, קבוצת פונקציות הניתנות לחישוב יעיל (פולינומי) היא זהה.

מחלקת POLY:

$POLY = \{f \mid f$ את המחשבת <u>פולינומית</u> <u>דטרמיניסטית</u> <u>קיימת מכונת טיורינג</u>$\}$

טענה

POLY סגורה להרכבה.

מכונת טיורינג אי-דטרמיניסטית

ההגדרה זהה להגדרת מכונת טיורינג דטרמיניסטית בשינוי יחיד: במקום פונקציית מעברים מגדירים <u>יחס</u>.

<u>בעץ חישוב:</u>

- שורש – הקונפיגורציה ההתחלתית
- עלה – קונפיגורציה סופית
- מסלול חישוב – מסלול מכוון משורש לעלה

מטא"ד M ***מקבלת*** קלט x אם <u>קיים</u> מסלול חישוב של M על x המסתיים בקונפיגורציה <u>מקבלת</u> (המכילה q_Y).

משפט מכונת טיורינג אי-דטרמיניסטית

המודל של מכונת טיורינג אי-דטרמיניסטית שקול חישובית למודל של מכונת טיורינג דטרמיניסטית.

מכונת טיורינג אי-דטרמיניסטית הינה ***פולינומית*** אם קיים פולינום $p(n)$ המהווה חסם סיבוכיות לכל ריצה אפשרית של המכונה (כלומר, לכל קלט ולכל מסלול חישוב עליו).

בעיה **ניתנת לפתרון** **בזמן פולינומיאלי** אם קיימת **מ"ט דטרמיניסטית** שרצה על כל קלט שלה בזמן פולינומיאלי .

בעיה **ניתנת לאימות** **בזמן פולינומיאלי** אם קיימת **מ"ט לא-דטרמיניסטית** שרצה על כל קלט שלה בזמן פולינומיאלי .

אלגוריתם אימות או **מאמת** לבעיה בוליאנית (בעית הכרעה), מקבל **קלט וניחוש** , ויודע לבדוק האם הניחוש הוא פתרון לבעיה עבור קלט זה .

מחלקת P

P היא מחלקת הבעיות הבוליאניות (בעיות הכרעה) הפתירות בזמן פולינומיאלי .

$P = \{ L \mid L$ קיימת מכונת טיורינג דטרמיניסטית פולינומית המקבלת את $L \}$

תכונות של P

P סגורה תחת: משלים, איחוד, חיתוך, שרשור

מחלקת NP

שפה L שייכת ל-***NP*** אם קיימת מכונת טיורינג אי-דטרמיניסטית פולינומית המקבלת את L.

NP היא מחלקת הבעיות הבוליאניות המאומתות בזמן פולינומיאלי .

טענה: $P \subseteq NP \subseteq R$

משפט (מרכזי במדעי המחשב)

$P = NP$ אמ"מ אימות יעיל גורר פתרון יעיל.

רדוקציה פולינומית מ- L_1 ל- L_2 היא פונקציה f המקיימת:

- f מלאה (מוגדרת על כל Σ^*).
- $f \in POLY$ (ניתנת לחישוב בזמן פולינומי)
- f מפרידה $x \in L_1 \iff f(x) \in L_2$

סימון: $L_1 \leq_p L_2$.

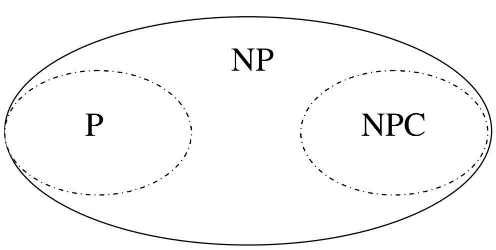
תכונות של רדוקציה פולינומית

- רפלקסיביות: $L \leq_p L$.
- טרנזיטיביות: $L_1 \leq_p L_3 \quad \Leftarrow \quad L_1 \leq_p L_2 , L_2 \leq_p L_3$.
- תקפות למשלים : $\overline{L_1} \leq_p \overline{L_2} \quad \Leftarrow \quad L_1 \leq_p L_2$.

משפט רדוקציה פולינומית

אם $L_1 \leq_p L_2$, אזי:

- אם $L_2 \in P$, אז גם $L_1 \in P$.
- אם $L_1 \notin P$, אז גם $L_2 \notin P$. (בהנחה ש- $P \neq NP$)



שפה L היא ***NP-קשה*** אם לכל $L' \in NP$ מתקיים $L' \leq_p L$.

שפה L היא ***NP-שלמה*** אם מתקיים:

- $L \in NP$.
- L היא NP-קשה.

NPC (*NP*-Complete) היא מחלקת שפות NP-שלמות.

משפט NPC

אם $L \in NPC$, אזי $P = NP \quad \Leftrightarrow \quad L \in P$.

משפט COOK-LEVIN

$$SAT \in \boldsymbol{NPC}$$

כמה שפות ב-NPC

$$SAT = \{\varphi \mid \varphi \text{ is satisfiable CNF formula}\}$$
$$3SAT = \{\varphi \mid \varphi \text{ is satisfiable 3CNF formula}\}$$
$$VC = \{(G,k) \mid G \text{ has a } k\text{-size vertex cover}\}$$
$$IS = \{(G,k) \mid G \text{ has a } k\text{-size independent set}\}$$

$$CLIQUE = \{(G,k) |\ G \text{ includes a } k\text{-size clique}\}$$
$$HAM\text{-}CYCLE = \{G |\ G \text{ is digraph and has a Hamiltonian cycle }\}$$
$$HAM\text{-}PATH = \{G |\ G \text{ is digraph and has a Hamiltonian path}\}$$
$$U\text{-}HAM\text{-}CYCLE = \{G |\ G \text{ is undirected graph and has a Hamiltonian cycle }\}$$
$$U\text{-}HAM\text{-}PATH = \{G |\ G \text{ is undirected graph and has a Hamiltonian path}\}$$
$$TSP = \{(G,k) |\ G \text{ is undirected complete weighted graph and has a Hamiltonian cycle with weight } \leq k\}$$
$$SSP = \{(A,s) |\ A \text{ is a set of natural numbers and } s \text{ is a sum of a subset of } A\ \}$$
$$PARTITION\ = \{A |\ A \text{ is a set of natural numbers that can be partitioned into two subsets with equal sums }\}$$
$$SETCOVER\ = \{(U,S,k) |\ U \text{ is a set, } S \text{ is a set of subsets of } U \text{ and } U \text{ is a union of } k \text{ subsets from } S\}$$

כמה שפות ב-P

קבוצת שפות חסרות הקשר מוכלת ב-**P** $(CFL \subset \mathbf{P})$

$$DNF\text{-}SAT = \{\varphi |\ \varphi \text{ is satisfiable DNF formula}\}$$
$$2SAT = \{\varphi |\ \varphi \text{ is satisfiable 2CNF formula}\}$$

משפט König

בגרף דו-צדדי גודל הכסוי הקודקודי המינימלי שווה לגודל הזווג המקסימלי.

תזכורות

DNF (disjunctive normal form) צורת ייצוג נוסחאות לוגיות (פסוקים) כסכום (לוגי) של מכפלות לוגיות (גוררים).

CNF (conjunctive normal form) צורת ייצוג נוסחאות לוגיות (פסוקים) כמכפלת הסכומים (פסוקיות).

נוסחא ספיקה (satisfiable) אם קיימת עבורה השמה מספקת (תחת ההשמה הנוסחא מקבלת ערך 1).

בצורת ***2CNF*** בכל פסוקית 2 ליטרלים בדיוק.

בצורת ***3CNF*** בכל פסוקית 3 ליטרלים בדיוק.

כיסוי ע"י צמתים (vertex cover, VC) בגרף $G(V,E)$ (לא מכוון) הוא תת-קבוצה $C \subseteq V$ המקיימת שלכל קשת $(v,u) \in E$ לפחות אחד הצמתים v,u שייך ל-C, כלומר $v \in C$ ו/או $u \in C$.

קבוצה בלתי תלויה (קב"ת, independent set, IS) בגרף $G(V,E)$ (לא מכוון) היא תת-קבוצה $S \subseteq V$ המקיימת שלכל זוג צמתים $v,u \in S$ לא קיימת קשת ביניהם, כלומר $(v,u) \notin E$.

גרף מלא (שלם) (complete graph) $G(V,E)$ (לא מכוון) הוא גרף בו לכל זוג צמתים $u,v \in V$ קיימת קשת ביניהם, כלומר $(v,u) \in E$.

קליק (תת-גרף מלא, clique) בגרף $G(V,E)$ (לא מכוון) הוא תת-קבוצה של צמתים $C \subseteq V$ המקיימת שלכל זוג צמתים $v,u \in C$ קיימת קשת ביניהם, כלומר $(v,u) \in E$.

מסלול/מעגל המילטוני (Hamiltonian path/cycle) בגרף $G(V,E)$ (מכוון/לא מכוון) הוא מסלול/מעגל שעובר בכל צמתי הגרף בדיוק פעם אחת.

TSP (Travelling Salesman Problem) בעיית הסוכן הנוסע (פה, בצורת בעיית הכרעה).

SSP (Subset sum problem) בעיית הסכום החלקי.

מחלקות משלימות

$$\boldsymbol{coP = P}$$
$$\boldsymbol{P \subseteq NP \cap coNP}$$
$$\boldsymbol{P = NP \Longrightarrow NP = coNP}$$

סיבוכיות מקום (נפח)

מכונת טיורינג הינה ***פולינומית במקום (זכרון)*** אם קיים פולינום $p(n)$ המהווה חסם על כמות תאי הסרט (הזכרון) שהמכונה משתמשת בהם במהלך הריצה לכל ריצה אפשרית של המכונה (כלומר, לכל קלט באורך n ולכל מסלול חישוב עליו).

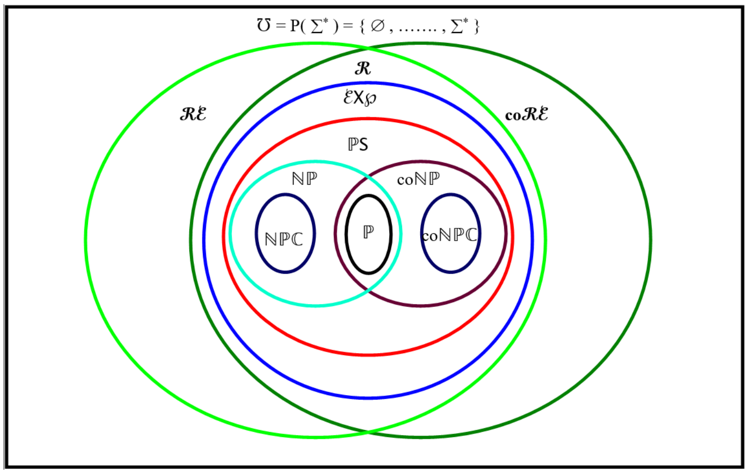
$\boldsymbol{PSpace} = \{\ L \mid$ קיימת מכונת טיורינג דטרמיניסטית פולינומית בזכרון המקבלת את $L\ \}$

$\boldsymbol{NPSpace} = \{\ L \mid$ קיימת מכונת טיורינג אי-דטרמיניסטית פולינומית בזכרון המקבלת את $L\ \}$

משפט SAVITCH

$\boldsymbol{PSpace = NPSpace}$

$$\boldsymbol{P \subseteq NP \subseteq PSpace \subseteq EXP \subseteq R}$$
$$\boldsymbol{coNP \subseteq coPSpace = PSpace}$$



אלגוריתמי קירוב

בעיית אופטימיזציה היא בעיית מציאת מינימום או מקסימום לאיזשהו ערך.

<u>נסמן:</u>

- S^* הוא פתרון אופטימלי לבעיה.
- S_A הוא פתרון המחושב ע"י אלגוריתם A.
- $P(S)$ הוא ערך הפתרון (רווח/עלות): ערך פתרון אופטימלי יסומן $P(S^*)$ (או OPT), ערך הפתרון המוחזר ע"י A יסומן $P(S_A)$ (או ALG).

בעיית מקסימום היא בעיית אופטימיזציה עבורה מחפשים פתרון מקסימלי.

נאמר כי אלגוריתם A עבור בעיית <u>מקסימום</u> מוצא פתרון אופטימלי עד כדי ***יחס קירוב כפלי*** $q \geq 1$, אם מתקיים $P(S_A) \geq \frac{P(S^*)}{q}$ במקרה הגרוע.

בעיית מינימום היא בעיית אופטימיזציה עבורה מחפשים פתרון מינימלי.

נאמר כי אלגוריתם A עבור בעיית <u>מינימום</u> מוצא פתרון אופטימלי עד כדי ***יחס קירוב כפלי*** $q \geq 1$, אם מתקיים $P(S_A) \leq q \cdot P(S^*)$ במקרה הגרוע.

קירוב בעיית *כיסוי בצמתים* MVC *(Minimum Vertex Cover problem)*

<u>בעיית האופטימיזציה</u>: בהנתן גרף למצוא בו כיסוי <u>מינימלי</u>.

אלגוריתם <u>יוריסטי</u> HVC (יחס קירוב לא קבוע)

<u>HVC</u> (G)
$C \leftarrow \varnothing$
$E' \leftarrow E$
while $E' \neq \varnothing$ **do**
  choose $v \in V$ s.t $d(v)$ is maximum // בחירת צומת עם דרגה מקסימלית
  $C \leftarrow C \cup \{v\}$ // הוספת הצומת לכיסוי
  for each $(v,x) \in E'$ **do** // מחיקת כל הקשתות המכוסות ע"י הצומת
    $E' \leftarrow E' \setminus \{(v,x)\}$
return C

אלגוריתם <u>קירוב</u> AVC (יחס קירוב כפלי 2)

<u>AVC</u> (G)
$C \leftarrow \varnothing$
$E' \leftarrow E$
while $E' \neq \varnothing$ **do**
  choose (arbitrarily) $(u,v) \in E'$ // בחירת קשת נוספת לשידוך
  $C \leftarrow C \cup \{u,v\}$ // הוספת קצות הקשת לכיסוי
  for each $(u,x) \in E'$ **do** // מחיקת כל הקשתות המכוסות ע"י הזוג הנ"ל
    $E' \leftarrow E' \setminus \{(u,x)\}$
  for each $(v,x) \in E'$ **do**
    $E' \leftarrow E' \setminus \{(v,x)\}$
return C

קירוב בעיית התרמיל KP *(Knapsack Problem)*

נתון ***תרמיל*** שלו קיבולת משקל W ונתון אוסף A של n פריטים. לכל פריט $a \in A$ מיוחס משקל $w(a)$ ורווח $p(a)$. יש לשמור על שלמות הפריטים. אין לשכפלם. *מילוי חוקי* של התרמיל הוא תת-קבוצה של אוסף הפריטים הנתון כך שהמשקל הכולל של הפריטים אינו עולה על קיבולת התרמיל.

<u>בעיית האופטימיזציה</u>: בהנתן תרמיל ואוסף פריטים למצוא מילוי חוקי עם רווח כולל <u>מקסימלי</u>.

אלגוריתם <u>חמדן</u> GKP (יחס קירוב לא קבוע)

<u>GKP</u> (W, A)

for each $a \in A$ **do** $u(a) \leftarrow \frac{p(a)}{w(a)}$ // חישוב רווח סגולי

sort A in a descending order (let a_i be an *i*-th element of *A* according to this order)

$F \leftarrow \varnothing$ // איתחול קבוצת מילוי

$s \leftarrow 0$ // משקל כולל של קבוצת המילוי

for each i **from** 1 **to** n **do** // מעבר על הפריטים לפי סדר לא עולה של u

  if $W - s \geq w(a_i)$ **do** // אם נותר מספיק מקום

    $F \leftarrow F \bigcup \{a_i\}$ // הוספת פריט למילוי

    $s \leftarrow s + w(a_i)$ // עדכון המשקל הכולל

return F

אלגוריתם קירוב AKP (יחס קירוב כפלי 2)

AKP (W, A)

$a_{max} \leftarrow \arg\max_{a \in A} \{ p(a) | \; w(a) \leq W \}$ // מציאת פריט יקר ביותר

if $p(a_{max}) \geq P(S_{GKP}(W, A))$ **return** $\{a_{max}\}$ // השוואתו עם הפתרון החמדני

return $S_{GKP}(W, A)$

קירוב בעיית הסוכן הנוסע *TSP* *(Traveling Salesman Problem)*

בעיית הסוכן הנוסע: נתון גרף (לא מכוון) מלא $G(V,E)$ בו לכל קשת $(v,u) \in E$ מוגדר משקל אי-שלילי $w(v,u)$. סימון: $G(V,E,w)$.

בעיית האופטימיזציה: בהנתן גרף מלא ממושקל למצוא בו מעגל המילטוני במשקל מינימלי.

אלגוריתם חמדן GTSP (יחס קירוב לא קבוע)

GTSP $(G(V,E,w))$

choose arbitrary vertex $v \in V$ // בחירת צומת שרירותי

$a \leftarrow v, \quad b \leftarrow v, \quad T \leftarrow V \setminus \{v\}, \quad H \leftarrow \varnothing$ // איתחולים

while $T \neq \varnothing$ **do** // לכל הצמתים שעוד לא במעגל

  choose $v \in T$ s.t $w(a,v)$ is minimum // בחירת צומת קרוב ביותר

  $T \leftarrow T \setminus \{v\}$ // סימון שהצומת כבר במעגל

  $H \leftarrow H \bigcup \{(a,v)\}$ // הוספת הקשת למעגל

  $a \leftarrow v$

$H \leftarrow H \bigcup \{(v,b)\}$ // סגירת המעגל

return H

משפט קירוב TSP

לא קיים (בהנחה ש- $P \neq NP$) אלגוריתם פולינומי המקרב את *TSP* עם יחס קירוב כפלי קבוע.

גרף ממושקל מלא (שלם) $G(V,E,w)$ הינו גרף ***מטרי*** אם לכל שלושה צמתים $x,y,z \in V$ מתקיים אי-שוויון המשולש $w(x,y) + w(y,z) \geq w(x,z)$.

בעיית הסוכן הנוסע מטרית: נתון גרף מטרי $G(V,E,w)$. יש למצוא בו מעגל המילטוני במשקל מינימום.

סימון: *MTSP* – Metric Traveling Salesman Problem.

אלגוריתם קירוב AMTSP (יחס קירוב כפלי 2)

AMTSP $(G(V,E,w))$

$T \leftarrow \text{Prim}(G(V,E,w))$ // מציאת עפ"מ

number all the vertices by running $DFS(T)$ (use $d(v)$) // מספור צמתים

let v_i be the vertex that $d(v) = i$ (by *DFS*)

$H \leftarrow \{(v_1,v_2),(v_2,v_3),(v_3,v_4),\ldots,(v_{|V|-1},v_{|V|}),(v_{|V|},v_1)\}$ // בניית המעגל

return H

קירוב בעיית החתך המקסימלי *Max-Cut*

בעיית האופטימיזציה: בהנתן גרף לא מכוון $G = (V,E)$ למצוא בו חתך מקסימלי.

אלגוריתם קירוב ל- Max-Cut (יחס קירוב כפלי 2)

קלט: גרף לא מכוון G = (V,E)

1. $S \leftarrow \emptyset,\ T \leftarrow V$.
2. כל עוד קיים קדקוד , שהעברתו מצד לצד תגדיל את החתך,
3. העבר אותו צד השני

פלט: S, T.

קירוב בעיית כיסוי בקבוצות המינימלי *Minimum Set-Covering*

בעית האופטימיזציה: בהינתן אוסף של קבוצות $S = \{S_1, S_2, S_3, \dots, S_k\}$ מעל עולם האיברים U (כך ש-$U = \bigcup_{i=1}^{k} S_i$), למצוא מספר מינימלי של הקבוצות המכסה את איברי U, כלומר: איחוד הקבוצות שנבחרו הוא U.

אלגוריתם קירוב ל- Minimum Set-Covering (יחס קירוב כפלי לא קבוע: $\ln|U| + 1$)

קלט: $S = \{S_1, S_2, S_3, \dots, S_k\}$ מעל U

1. $C \leftarrow \emptyset,\ X \leftarrow U,\ F \leftarrow S$.
2. כל עוד $X \neq \emptyset$
3. בחר קבוצה $S_i \in F$ עבורה $|S_i \cap X|$ מקסימלי
4. $C \leftarrow C \cup \{S_i\},\ X \leftarrow X \setminus S_i,\ F \leftarrow F \setminus \{S_i\}$

פלט: C.

קירוב בעיית האריזה *Bin-Packing*

בעית האופטימיזציה: בהינתן אוסף של n פריטים $S = \{a_1, a_2, a_3, \dots, a_n\}$ כך ש-$0 < a_i \le 1$ $\forall 1 \le i \le n$, למצוא מספר מינימלי של אריזות **יחידה**, בהן ניתן לארוז את הפריטים.

אלגוריתם קירוב First-Fit ל- Bin-Packing (יחס קירוב כפלי 2)

קלט: $S = \{a_1, a_2, a_3, \dots, a_n\}$

1. $B \leftarrow \emptyset,\ k \leftarrow 0$.
2. לכל i החל מ-1 וכלה ב-n :
3. אם קיימת אריזת יחידה $B_j \in B$ שניתן להוסיף לה את a_i
4. $B_j \leftarrow B_j \cup \{a_i\}$
5. אחרת
6. $B_{k+1} \leftarrow \{a_i\},\ B \leftarrow B \cup \{B_{k+1}\},\ k \leftarrow k+1$

פלט: B.

אלגוריתמים מקוונים

אלגוריתם מקוון (on-line) היא בעית מציאת מינימום או מקסימום לאיזשהו ערך.

- הקלט לא ידוע מראש – הוא מגיע בשלבים .
- בכל פעם שמתקבל חלק מהקלט , האלגוריתם מגיב ומבצע משהו .
- כשהאלגוריתם מקבל איבר קלט , הוא לא יודע מה יהיו איברי הקלט אחריו .

נאמר כי אלגוריתם A עבור בעית מקסימום משיג ***יחס תחרותיות*** $R \ge 1$, אם מתקיים $P(S_A) \ge \frac{P(S^*)}{R}$ במקרה הגרוע $\left(ALG \ge \frac{OPT}{R}\right)$.

נאמר כי אלגוריתם A עבור בעית מינימום משיג ***יחס תחרותיות*** $R \ge 1$, אם מתקיים $P(S_A) \le R \cdot P(S^*)$ במקרה הגרוע $(ALG \le R \cdot OPT)$.

בעיית הסקי *(Ski rental problem)*

חופשת סקי מתבצעת בצורה רצופה מספר ימים. נסמן את מספר הימים ב-D, אך אינו ידוע מראש (הודעת סיום החופשה מגיעה ביום הסיום). עלות קנית ציוד הסקי היא מספר שלם M>1, אך ניתן לשכור ציוד ליום במחיר 1. נדרש להחליט האם לשכור את הציוד כל יום או לקנות, ומתי (באיזה יום של החופשה), ע"מ למזער את העלות הכוללת.

אלגוריתם אופטימלי (off-line)

כאשר D ידוע: אם M>D, אז כדאי לשכור את הציוד כל יום, אחרת – לקנות. העלות האופטימלית היא $P(S^*) = \min\{D, M\}$.

אלגוריתם on-line ל-Ski rental (יחס תחרותיות 2)

לקנות ביום M (אם החופשה לא נגמרת קודם), עד אז – לשכור. העלות במקרה הגרוע היא $P(S_A) = 2M$.

בעיית השועל והכרם

השועל מעוניין למצוא את הפרצה בגדר המקיפה את הכרם. הגדר היא אינסופית. הפרצה קיימת בוודאות, אך יכולה להימצא מימין לשועל או משמאלו במרחק D (לא ידוע מראש).

אלגוריתם אופטימלי (off-line)

כאשר ידוע הכיוון: ללכת בכיוון הידוע עד לפרצה. העלות האופטימלית היא $P(S^*) = D$.

אלגוריתם Fox-light on-line לבעיית השועל והכרם (גרסה מוקלת, יחס תחרותיות 3)

כאשר ידוע המרחק D (אך לא ידוע הכיוון): ללכת בכיוון שרירותי למרחק D. אם לא נמצאה הפרצה, לחזור לנקודת ההתחלה וללכת בכיוון השני עד לפרצה.
העלות במקרה הגרוע היא $P(S_A) = 3D$.

אלגוריתם Fox-full on-line לבעיית השועל והכרם (גרסה מלאה, יחס תחרותיות 9)

1. בחר כיוון שרירותי.
2. $a = 1$
3. לך בכיוון הנבחר למרחק a
4. אם נמצאה הפרצה, סיים
5. אחרת:
6. הפוך את הכיוון
7. לך בכיוון הנבחר למרחק a (לחזור לנקודת ההתחלה)
8. $a \leftarrow 2a$
9. חזור ל-(3)

העלות במקרה הגרוע היא $P(S_A) \leq 9D$.

בעיית ***הדפדוף*** ***(Paging problem)***

נתון דיסק חיצוני גדול ואיטי המכיל n דפים וזכרון פנימי (מטמון, cache) קטן ומהיר המכיל k דפים (k<<n). הקלט הוא סדרת בקשות לדפים מהדיסק (לא ידועה מראש). אם הדף המבוקש לא נמצא במטמון, יש להביאו מהדיסק – פעולה יקרה. ואם באותו רגע המטמון מלא, כתיבת הדף המובא תדרוס אחד הדפים הקיימים במטמון. המטרה היא למלא את כל הבקשות לדפים לפי הסדר תוך מספר מינימלי של הבאות דפים מהדיסק.

לשם ניתוח יחס התחרותיות, נחלק כל קלט נתון (סדרת בקשות), החל מהבקשה הראשונה, לתתי-סדרות $S = S_1,S_2,\ldots,S_t$,
כך שכל תת-סדרה תהיה המקסימלית (באורכה) המכילה k דפים שונים. נתייחס ל-t – מס' תתי-הסדרות הנ"ל.

אלגוריתם <u>אופטימלי</u> (off-line)

כאשר ידועה סדרת הבקשות: כל פעם כשצריך להסיר דף מה-cache המלא, בוחרים דף שהבקשה הבאה שלו היא כמה שיותר רחוקה בעתיד. העלות האופטימלית תלויה בסדרת הבקשות. במקרה הטוב ביותר, $P(S^*) \geq t$.

אלגוריתם FWF on-line (flush when full) (יחס תחרותיות k)

מרוקן את ה-cache בתחילת כל סדרה חדשה, ומשלם על כל תת-סדרה k (פרט אולי לסדרה האחרונה). העלות במקרה הגרוע היא $P(S_A) \leq k \cdot t$.

אלגוריתם FIFO on-line (first in first out) (יחס תחרותיות k)

כל פעם כשצריך להסיר דף מה-cache המלא, נבחר הדף הישן ביותר (עם זמן ההכנסה הכי מוקדם). העלות במקרה הגרוע היא $P(S_A) \leq k \cdot t$.

אלגוריתם LRU on-line (least recently used) (יחס תחרותיות k)

כל פעם כשצריך להסיר דף מה-cache המלא, נבחר דף שהשימוש האחרון בו היה לפני פרק זמן גדול ביותר. העלות במקרה הגרוע היא $P(S_A) \leq k \cdot t$.

בעיית ***המזכירה (המנהלת)*** ***(Secretary problem)***

מנהל מראיין n מועמדות לתפקיד מזכירה. לאחר כל ראיון עליו להחליט אם לבחור במועמדת הנוכחית או לדחותה (מבלי אפשרות לפנות אליה בהמשך). מטרת המנהל היא לבחור את המועמדת הטובה ביותר.

אלגוריתם <u>אופטימלי</u> (off-line)

כאשר ניתן לראיין את כל המועמדות לפני קבלת ההחלטה: ע"ס כל הראיונות לבחור את הכי טובה.

אלגוריתם on-line הסתברותי

ראיין ודחה את k המועמדות הראשונות, ואז בחר במועמדת הבאה, שהיא טובה יותר מכל k המועמדות הראשונות.
אם אין אחת כזאת , בחר במועמדת האחרונה .

טענה של מרטין גרדנר

בחירת $k = \frac{n}{e} \approx 0.37n$ מביאה למקסימום (~0.36788) את ההסתברות לבחירת המועמדת המוצלחת ביותר.

KMP-MATCHER(T, P)

```
n = T.length
m = P.length
π = COMPUTE-PREFIX-FUNCTION(P)
q = 0                                   // number of characters matched
for i = 1 to n                          // scan the text from left to right
    while q > 0 and P[q + 1] ≠ T[i]
        q = π[q]                        // next character does not match
    if P[q + 1] == T[i]
        q = q + 1                       // next character matches
*   τ[i] = q
    if q == m                           // is all of P matched?
        print "Pattern occurs with shift" i − m
        q = π[q]                        // look for the next match
```

COMPUTE-PREFIX-FUNCTION(P)

```
m = P.length
let π[1 . . m] be a new array
π[1] = 0
k = 0
for q = 2 to m
    while k > 0 and P[k + 1] ≠ P[q]
        k = π[k]
    if P[k + 1] == P[q]
        k = k + 1
    π[q] = k
return π
```

<u>אלגוריתם KMP:</u>

<u>קלט</u>: תבנית P וטקסט T

1. נבנה את המערך state על ידי הפונקציה SetState.
2. נריץ את האוטומט עם הטקסט T כקלט בעזרת הפונקציה Search ונשתמש בפונקציית המעברים δ שכתבנו.

```
SetState(String p[1...m])     // Fill state array.
state[1] ← 0
for i ← 1,...,m-1 do
    state[i+1] ← δ(state[i], p[i+1])
```

```
δ(State i, Char c)
if c = p[i+1] then
    return i + 1
else if i = 0 then
    return 0
else
    return δ(state[i],c)
```

```
Search(char T[1..n], Transition Function  δ)
s ← 0       // initial state
for i ← 1,...,n do
    s ← δ(s, T[i])
    if s = m then
        s=state(m)
        print Match in position i − m + 1
```