BFS

נותן פיתרון ל:
- מציאת מסלול קצר ביותר מהמקור לכול צומת.
- גילוי צמתים ברי הגעה מהמקור.

האלגוריתם:
```
for each u ∈ V–{s} :
        d[u] = ∞
        p[u] = null

Q = EMPTY
enqueue(Q,s)

while Q is not EMPTY do:
        u = dequeue(Q)
        for each v in adjacent(u) do:
                if d[v] = ∞ then:
                        d[v] = d[u] +1
                        p[v] = u
                        enqueue(Q,v)
```

תכונות:
- סיבוכיות זמן O(V+E).
- בתור Q יש צמתים משתי רמות לכול היותר שמסודרים בסדר לא יורד.

שיטות יישום:
- שכפול הצמתים והקשתות בהתאם לתכונתם.
- אם יש מצבים לא חוקיים בהליכה על מסלול ניתן לשכפל את הצמתים כמספר המצבים החוקיים ולבנות קשתות כך שמעבר עליהם יגרום להחלפת מצב חוקי בנוסף למעבר לצומת הבאה.

DFS

נותן פיתרון ל:
- מציאת מעגלים בגרף.
- מציאת שורשים בגרף.
- כיוון קשתות הגרף.

האלגוריתם:
```
for each u ∈ V :
        color[u] = white
        p[u] = null

i = 0
for each u ∈ V :
        if color(u) = white then:
        DFS_VISIT(u)

DFS_VISIT(u):
        color(u) = gray
        i++
        d[u] = i
        for each v in adjacent(u):
                if color(v) = white then:
                        p[v]=u
                        DFS_VISIT(v)
        color(u) =  black
        i++
        f[u] = i
```

תכונות:
- סיבוכיות זמן O(V+E).
- קשת עץ DFS – מצומת אפור לצומת לבן.
- קשת אחורית – מצומת אפור לצומת אפור.
- קשת קדמית – מצומת אפור לצומת שחור.
- קשת חוצה – מצומת אפור לצומת שחור.
- בגרפים לא מכוונים יש רק קשתות עץ או אחוריות.
- כול קשת אחורית סוגרת מעגל.
- צומת v צאצא של u אם"ם בזמן גילוי u יש מסלול מ u ל v המורכב כולו מצמתים לבנים.
- לכול זוג צמתים u,v יש ייחס אב קדמון צאצא אם"ם האינטרוול (d[v],f[v]) מוכל ב (d[u], f[u])

מיון טופולוגי של גרף מכוון חסר מעגלים

נותן פיתרון ל:
- מציאת סדר טופולוגי בין הצמתים.

האלגוריתם:
- מצא בעזרת DFS את ערכי f[v] לכול צומת.
- סדר המיון הטופולוגי הוא סדר הפוך של ערכי f[v]

תכונות:
- סדר טופולוגי קובעה יחסי אב קדמון-צאצא (או חוסר ייחס) בין צמתי הגרף.

מציאת רכיבים קשרים היטב (SCC)

הגדרות:
- רכיב קשיר היטב מהווה קבוצה מקסימאלית של צמתים שכנים להם יש מסלול מכוון מכול צומת לכול צומת בקבוצה.

אלגוריתם למציאה:
- מצא בעזרת DFS את ערכי f[v] לכול צומת.
- הפוך את כיווני הקשתות בגרף המקורי.
- הפעל DFS על הגרף ההפוך אבל בחר את הצמתים בלולאה הראשית לפי סדר יורד של f[v]
- כול עץ DFS שמתקבל הוא רכיב קשיר היטב.

תכונות:
- סיבוכיות זמן O(V+E).
- גרף המורכב מרכיבי קשירות הוא גרף מכוון חסר מעגלים (DAG).

מציאת רכיבים אי פריקים

הגדרות:
- רכיב אי פריק מהווה קבוצה מקסימאלית של צמתים שכנים שהסרת אחד מהם לא תפגע בקשירות הקבוצה .
- צומת הפרדה הוא צומת בודד שהסרתו תפגע בקשירות הגרף.
- גרף רכיבים אי פריקים מורכב מצמתי הפרדה ומצמתים המייצגים רכיבים אי פריקים. ישנן קשתות רק בין רכיבים אי פריקים לצמתי הפרדה המוכלים בהם. גרף זה הוא עץ.

אלגוריתם למציאה:
(ואריאציה על DFS)

תכונות:
- סיבוכיות זמן O(E).
- לכול צומת הפרדה יש לפחות זוג אחד של צמתים בגרף כך שהוא נמצא על כול המסלולים בינם.
- לכול רכיב אי פריק שאינו גשר (יותר מ 2 צמתים) כול צמתיו יושבים על מעגל פשוט אחד או יותר.

עץ פורש מינימום (עפ"מ)

הגדרות:
- עץ פורש בעל משקל קשתות מינימאלי.
- כלל כחול – בחר חתך חסר קשתות כחולות וצבעה בכחול קשת לא צבועה הקלה ביותר בחתך.
- כלל אדום – בחר מעגל חסר קשתות אדומות וצבעה באדום קשת לא צבועה הכבדה ביותר.

האלגוריתם הגנרי:
- כול עוד יש קשתות לא צבועות בגרף הפעל את הכללים הכחול ואדום באופן שרירותי.
- כול הקשתות הכחולות מגדירות עפ"מ.

תכונות:
- לכול עפ"מ יש סדרת הפעלות דטרמיניסטי של הכללים שמוצאה אותו.

אלגוריתם של Prim למציאת עפ"מ

האלגוריתם:
- תגדיר עץ T והכנס לתוכו צומת כלשהו בגרף.
- כול עוד יש קשתות לא צבועות: בחר קשת קלה ביותר בחתך בין T לשאר הגרף וצבעה אותה בכחול, הכנס את הצומת v שלה לעץ וצבעה באדום את כול הקשתות האחרות בחתך שמובילות ל v.
- כול הקשתות הכחולות מגדירות עפ"מ.

תכונות:
- סיבוכיות זמן O(E*log(V)) במימוש עם ערימה רגילה וניתן לקבל O(E+V*log(V)) במימוש עם ערימת פיבונאצ'י.
- מקרה פרטי של האלגוריתם הגנרי.

אלגוריתם של Kruskal למציאת עפ"מ

האלגוריתם:
- מיין את הקשתות לפי משקל בסדר לא יורד.
- לכול קשת לפי סדר המיון, אם היא לא סוגרת מעגל של קשתות כחולות צבעה אותה בכחול, אחרת באדום.
- כול הקשתות הכחולות מגדירות עפ"מ.

תכונות:
- סיבוכיות זמן O(E *log(V))
- מקרה פרטי של האלגוריתם הגנרי.

שיטות יישום:
- ניתן להגדיר סדר מיון בין קשתות ממשקל זהה לפי תכונה אחרת כמו צבע, מה שייתן עדיפות לקשתות אלה וימצא עפ"מ שיכיל כמה שיותר מהם (עפ"מים הצהובים ביותר).

מסלולים קלים ביותר

הגדרות:
- מסלול קל ביותר מצומת s ל v הוא מסלול שמשמקלו, dist(s,v) מינימאלי מבין המסלולים.
- פונקצית חסם עליון על צומת, d(v) מקיימת ש d(v) ≥ dist(s,v).
- שיפור מקומי על קשת (u,v) מתבצע אם d(v) > d(u)+w(u,v) אז d(v) ←d(u)+w(u,v)

אלגוריתם גנרי:
- קבע פונקצית חסם עליון חוקית על הצמתים.
- כול עוד יש קשת עליה ניתן לבצע שיפור מקומי בצעה שיפור.
- כשאין קשת עליה ניתן לבצע שיפור מקומי אז לכול צומת מתקיים d(v) = dist(s,v).

תכונות:
- מסלול קל ביותר מכיל לא יותר מ |V|-1 קשתות.
- על גרף בעל קשתות הפוכות נמצאה את המסלול הקל ביותר מכול צומת למקור.

עץ מסלולים קלים ביותר

הגדרות:
- עץ מסלולים קלים ביותר הוא תת גרף שהוא עץ בו המסלול היחיד לכול צומת מהמקור הוא המסלול הקל ביותר בגרף.

אלגוריתם למציאה:
- בנה תת גרף המורכב מכול הצמתים ומאיחוד הקשתות על המסלולים הקלים ביותר לכול צומת.
- הרץ BFS על תת הגרף מצומת המקור.
- החזר את עץ BFS הנוצר מתת הגרף.

תכונות:
- עץ מסלולים קלים של גרף נשאר זהה אם משנים את המשקל על הקשתות ע"י הכפלה בגורם קבוע.
- עץ מסלולים קלים של גרף משתנה אם משנים את המשקל על הקשתות ע"י הוספת גורם קבוע.

מציאת מסלולים קלים ביותר לפי Belman-Ford

האלגוריתם:
- קבע d(s)=0 ולשאר הצמתים d(v)= ∞
- חזור |V|-1 פעמים:
  - בצע שיפור מקומי לכול קשת בגרף.

תכונות:
- סיבוכיות זמן O(VE).
- לכול צומת v בעל מסלול קל ביותר מ s המכיל k קשתות, אחרי k איטרציות, d(v) =dist(s,v).
- אם באיטרציה ה |V| ניתן לעשות שיפור מקומי אז יש בגרף מעגלים שליליים.

מציאת מסלולים קלים ביותר לפי Dijkstra

האלגוריתם:
- קבע d(s)=0 ולשאר הצמתים d(v)= ∞
- Q ← V
- כול עוד Q לא ריק:
  - הוצא v בעל d(v) מינימאלי.
  - לכול קשת (v,u) בצע שיפור מקומי.

תכונות:
- סיבוכיות זמן O(E*log(V)) במימוש עם ערימה רגילה וניתן לקבל O(E+V*log(V)) במימוש עם ערימת פיבונאצ'י.
- נכון רק לגרף ללא משקלים שליליים.

אלגוריתם של Johnson

נותן פיתרון ל:
- מציאת מסלולים קלים ביותר בין כול שני צמתים, בגרף בעל קשתות עם משקלים שליליים.

האלגוריתם:
- הוסף צומת s וקשת ממנו במשקל 0 לכול צומת.
- הרץ Belman-Ford מ s וודע שאין מעגל שלילי.
- לכול קשת הגדר פונקצית משקל חדשה: w'(u,v)= w(u,v)+d(u)−d(v)
- הרץ Dijkstra מכול צומת בגרף.
- תקן לכול צומת את משקל המסלול הקצר: d (u,v)= d'(u,v)-d(u)+d(v)

תכונות:
- סיבוכיות זמן O(V(E+V*log(V))).

אלגוריתם של Floyd–Warshall (תכנות דינאמי)

נתון פיתרון ל:
- מציאת מסלולים קלים ביותר בין כול שני צמתים, בגרף בעל n צמתים וקשתות עם משקלים שליליים.

האלגוריתם:

$$\begin{array}{l}
for\,each\,[(i,j)/\ i\neq j]\,do:\\
\quad if\,[(i,j)\in E]\,then:\,d(i,j)=w(i,j)\\
\quad else:\,d(i,j)=\infty\\
\\
for[k=1\,to\,n]do:\\
\quad for\,each\,[(i,j)/\ i\neq j]\,do:\\
\quad\quad d(i,j)=\min[d(i,j)\ ,\ d(i,k)+d(k,j)]
\end{array}$$

תכונות:
- סיבוכיות זמן $O(V^3)$.

קבוצת קטעים בלתי תלויה במשקל מקסימאלי

הגדרות:
- קבוצת קטעים מכילה איברים $\{a_1,\ldots,a_n\}$ המאופיינים ע"י זמן התחלה s_i זמן סיום f_i ומשקל w_i.
- תת קבוצה בלתי תלויה במשקל מקסימאלי של קטעים מקיימת שהקטעים זרים בזוגות ומשקלם הכולל הוא מקסימאלי אפשרי. נגדיר $opt(i)$ להיות תת קבוצה זו מתוך i האיברים הראשונים בקבוצה.
- נגדיר $pred(i)$ להיות האינדקס של הקטע שמסתיים הכי מאוחר מבין כול הקטעים שמסתיימים לפני ש i מתחיל.

אלגוריתם למציאה:

$$\begin{array}{l}
sort(S)\ by\ f_i\\
opt(0)=0\\
A\leftarrow\varnothing\\
\\
for[i=1\ \ to\ n]\ do:\\
\quad if\,[opt(i-1)<opt(pred(i))+w_i]\ do:\\
\quad\quad A(i)\leftarrow A(pred(i))\cup a_i\\
\quad\quad opt(i)=opt(pred(i))+w_i\\
\quad else:\\
\quad\quad A(i)\leftarrow A(i-1)\\
\quad\quad opt(i)=opt(i-1)\\
return\ A(n)
\end{array}$$

תכונות:
- סיבוכיות זמן O(n*log(n))
- אלגוריתם ממש את הפונקציה:

$$opt(i)=\max\begin{Bmatrix}opt(i-1)\\opt(pred(i))+w_i\end{Bmatrix}$$

סידור אופטימאלי לכפל של מטריצות

הגדרות:
- בהינתן n מטריצות כך שלכול i מימד המטריצה הוא $p_{i-1}\times p_i$ נמצא את מספר המכפלות המינימאלי $m[1,n]$ הדרוש לביצוע כפל המטריצות.

אלגוריתם למציאה:

$$\begin{array}{l}
for[i=1\,to\,n]\,do:\ m[i,i]=0\\
for[j=1\,to\,n-1]\,do:\\
\quad for[l=1\,to\,n-j]\,do:\\
\quad\quad m[l,l+j]=\\
\quad\quad \min_{l\le k<l+j}\{m[l,k]+m[k+1,l+j]+p_{l-1}p_k\,p_{l+j}\}
\end{array}$$

תכונות:
- סיבוכיות זמן $O(n^3)$
- אלגוריתם ממש את הפונקציה:

$$m[i,j]=\begin{Bmatrix}0 & i=j\\ \min\limits_{i\le k<j}\{m[i,k]+m[k+1,j]+p_{i-1}p_k p_j\} & \end{Bmatrix}$$

מציאת אורך תת מחרוזת משותפת ארוכה ביותר

הגדרות:
- תת מחרוזת היא מחרוזת המורכבת מתת קבוצה של תווים (לא בהכרח עוקבים) במחרוזת נתונה, המסודרים לפי אותו סדר.
- תת מחרוזת משוטפת Z מוגדרת עבור זוג מחרוזות נתון X, Y באורכים m ו n בהתאמה, כך ש Z היא תת מחרוזת של X ושל Y.
- אורך תת מחרוזת משותפת ארוכה ביותר של מחרוזות באורך m , n מסומן כ $c(m,n)$

האלגוריתם:

$$\begin{array}{l}
for[i=1\,to\,m]\,do:c(i,0)=0\\
for[j=1\,to\,n]\,do:c(0,j)=0\\
for[i=1\,to\,m]\,do:\\
\quad for[j=1\,to\,n]\,do:\\
\quad\quad if\ [x_i=y_j]then:\\
\quad\quad\quad c(i,j)=c(i-1,j-1)+1\\
\quad\quad else:\\
\quad\quad\quad c(i,j)=\max\{c(i-1,j),c(i,j-1)\}
\end{array}$$

תכונות:
- סיבוכיות זמן O(m*n).
- אלגוריתם ממש את הפונקציה:

$$c(i,j)=\begin{cases}0 & i=0\ or\ \ j=0\\ c(i-1,j-1)+1 & i,j>0, x_i=y_j\\ \max\{c(i,j-1),c(i-1,j)\} & i,j>0, x_i\neq y_j\end{cases}$$

- מהמטריצה c ניתן למצוא את תת המחרוזת עצמה ב O(m+n).

מציאת קבוצת צמתים בלתי תלויה מקסימאלית בעץ

הגדרות:
- בעץ T קבוצת צמתים בלתי תלויה מורכבת מצמתים שאין קשת בין אף אחד מהם.

האלגוריתם:

$$\begin{array}{l}
for\,each[v]in\,preorder[T]:\\
\quad S^+(v)=1+\sum\limits_{u\in children(v)}S^-(u)\\
\quad S^-(v)=\sum\limits_{u\in children(v)}\max\{S^+(u),S^-(u)\}\\
\\
return\ \max\{S^+(root),S^-(root)\}
\end{array}$$

תכונות:
- סיבוכיות זמן O(d(v)*V).
- באותה סיבוכיות ניתן למצוא את הצמתים בקבוצה ע"י מעבר על הערכים שחושבו באלגוריתם.

בעיית הצביעה של גרפים

הגדרות:
- צביעה חוקית של גרף היא התאמה של צבע לכול צומת כך שכול זוג צמתים שכנים צבועים בצבעים שונים. אופטימאלית משתמשת במינימום צבעים.

אלגוריתם חמדן:
- לפי סדר כלשהו לכול צומת בגרף:
  - צבע את הצומת בצבע חוקי הקטן ביותר.

תכונות:
- סיבוכיות זמן O(V+E).
- משתמש במספר צבעים לא גדול מדרגה מקסימאלית של צומת בגרף + 1.
- אופטימאלי אם הגרף הוא קליק או מעגל אי זוגי.
- לכול צביעה אופטימאלית קיים סידור של צמתים לפיו האלגוריתם החמדן נותן אתה.

שיטות יישום:
- אם בוחרים סדר צביעה כך שלכול צומת שצובעים כול השכנים הצבועים שלו מרכיבים קליק אז הצביעה הכללית נותנת תוצאה אופטימאלית.
- גרף אינטרוולים נצבעה אופטימאלית אם בוחרים סדר צביעה לפי זמני התחלה (מקטן) של אינטרוול

קידוד Huffman בקוד σ-ארי

נתון פיתרון ל:
- בהינתן קבוצת תווים W (|W|=n) ופונקצית תדירות f(W)→R, נקבל עץ T σ-ארי המגדיר קידוד התווים למילים מעל א"ב σ-ארי כך שאורך הקוד יהיה מינימאלי.

האלגוריתם:

$$\begin{array}{l}
Huffman(W):\\
if\ \ |W|=1\ return\ T\\
repeat\ [\ (n\text{-}2)\bmod(\sigma\text{-}1)\text{+}2\ ]\ times:\\
\quad W_{\min}\leftarrow PopLeastFrequent(W)\\
CreateNewWord(w')\\
f(w')=\sum f(W_{\min})\\
W'=(W\setminus W_{\min})\cup\{w'\}\\
T=Huffman(W')\\
AddChildren(w',W_{\min})\\
return\ T
\end{array}$$

תכונות:
- סיבוכיות זמן O(n*log(n)).
- העץ מגדיר קוד פרפיקסי.
- בעץ σ-ארי מלא עם k צמתים פנימיים מתקיים $n=(\sigma-1)k+1$
- כול הצמתים הפנימיים בעץ מלאים (חוץ מאחד).
- מספר העלים לצומת הלא מלא היחיד בעץ הוא $(n-2)\bmod(\sigma-1)+2$ והם בעלי התדירות המינימאלית בעץ.
- אורך ממוצע של מילת קוד תמיד גדול מ log(n).

רשתות זרימה

נתון פיתרון ל:
- בהינתן גרף מכוון G בעל צומת מקור s , צומת בור t ופונקצית קיבול c(e) על הקשתות, נמצא פונקצית זרימה חוקית מקסימאלית.

הגדרות:
- פונקצית זרימה חוקית F מגדירה לכול קשת ערך זרימה f(e) המקיים $0\le f(e)\le c(e)$, ולכול צומת חוץ מהמקור והבור מתקיים שסכום ערכי הזרימה על הקשתות הנכנסות שווה לסכום ערכי הזרימה על הקשתות היוצאות.
- חתך בגרף מוגדר ע"י חלוקת הגרף לשתי קבוצות זרות של צמתים וכול קשת מצומת בקבוצה אחת לאחרת היא קשת החתך, קיבול החתך הוא סכום הקיבולים על קשתות אלה, והחתך הוא חתך מינימום אם סכום זה הוא מינימאלי.
- גודל הזרימה ניתן לחשב ע"י נטו הזרימה מצומת המקור או לצומת בור או זרימה דרך כול חתך.
- גרף שיורי הוא גרף בו כול קשת מקורית e הופכת לשתי קשתות הפוכות – קדמית בקיבול c(e)–f(e) ואחורית בקיבול f(e) (קשתות בקיבול 0 מוסרות).
- מסלול שיפור הוא מסלול מ s ל t בגרף השיורי.
- אם Δ זה הקיבול השיורי המינימאלי על מסלול שיפור, אז נגדיר שיפור על מסלול שיפור ע"י הגדלת הקיבול ב Δ על כול קשת קדמית במסלול והקטנת הקיבול ב Δ על כול קשת אחורית במסלול.

אלגוריתם גנרי (Ford–Fulkerson):
- לכול קשת בגרף קבע f(e)=0
- כל עוד קיים מסלול שיפור בגרף השיורי שפר עליו את הזרימה.

תכונות:
- סיבוכיות זמן בקיבולים שלמים O(|F|*E).
- אם הקיבולים הם בשלמים אז האלגוריתם ימצא זרימת מקסימום בשלמים ויעצור.
- כול התנאים הבאים שקולים:
  - F היא זרימת מקסימום.
  - אין מסלול שיפור בגרף השיורי.
  - קיים חתך שקיבולו שווה לגודל הזרימה, חתך זה הוא חתך מינימום בגרף.
- איחוד או חיתוך של כול חתכי המינימום הוא גם חתך מינימום.

אלגוריתם של Edmonds–Karp:
- מוצא זרימת מקסימום ב $O(E^2V)$.

אלגוריתם של Dinic:
- מוצא זרימת מקסימום ב $O(EV^2)$.

מציאת שידוך מקסימום בגרף דו צדדי

הגדרות:
- שידוך מקסימום בגרף זו תת קבוצה מקסימאלית של קשתות כך שלכול שתי קשתות בתת קבוצה אין צומת משותפת. השידוך הוא מושלם כאשר הוא מחסה את כול צמתי הגרף.

אלגוריתם למציאה:
- הוסף לגרף דו צדדי צומת מקור וצומת בור וקשתות מצומת מקור לכול צמתי צד שמאל, ומכול צמתי צד ימין לצומת בור.
- קבע קיבולים שלמים של 1 על כול הקשתות.
- מצא זרימת מקסימום בגרף והחזר את כול הקשתות בין צד שמאל לצד ימין בהם יש זרימה של 1.

תכונות:
- בגרף דו צדדי יש שידוך מגודל k אם"ם יש לו זרימה מגודל k.
- בגרף דו צדדי יש שידוך מושלם אם"ם מתקיים שלכול תת קבוצה של צמתי צד שמאל, קבוצת הצמתים השכנים לה בצד ימין לא קטנה ממנה.

מאת אולג קברניק