Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Wiktor Zuba
Nr albumu: 320501

Efektywne algorytmy generacji obiektów kombinatorycznych???

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
prof. Wojciech Rytter
Instytut Informatyki

??? 2017

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data | Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data | Podpis autora (autorów) pracy

Streszczenie

???

Słowa kluczowe

???

Dziedzina pracy (kody wg programu Socrates-Erasmus)

???

Klasyfikacja tematyczna

???

Tytuł pracy w języku angielskim

Effective algorithms of combinatorial objects generation???

Spis treści

Wprowadzenie

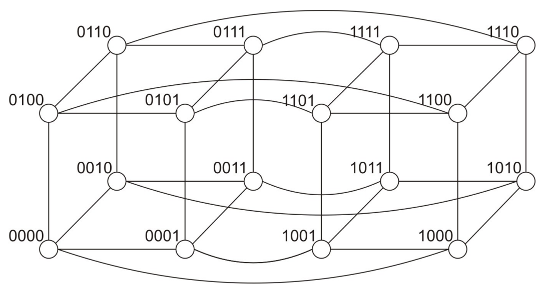
Rozdział 1

Własności hiperkostki

1.1. Podstawowe definicje

Definicja 1.1.1. *Dla* $n \in \mathbb{N} \quad [n] = \{0, ..., n-1\}$ *(zbiór pierwszych* n *liczb naturalnych).*

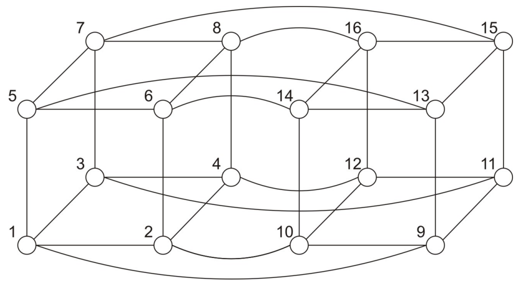
Definicja 1.1.2. Hiperkostką wymiaru n (Q_n) *nazwiemy graf, w którym każdy wierzchołek odpowiada ciągowi binarnemu długości* n*, zaś krawędzią połączone są te wierzchołki, których ciągi binarne różnią się na dokładnie jednej pozycji.*
$V(Q_n) = \{(v_0, ..., v_{n-1}) : v_i \in \{0,1\}\}, E(Q_n) = \{(u,v) : \sum_i |u_i - v_i| = 1\}$



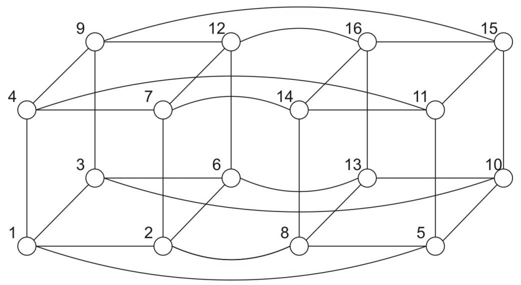
W przypadku pełnej hiperkostki bardzo łatwo jest określić długość najkrótszej ścieżki pomiedzy wierzchołkami – jest ona równa ilości pozycji na których różnią się ciągi tych wierzchołków.
Hiperkostka jest grafem dwudzielnym, w którym jedną składową jest zbiór wierzchołków o ciągach z parzystą liczbą jedynek, zaś drugą tych o ich nieparzystej liczbie.
Cowięcej przy badaniu hiperkostek często dzieli się je na $n+1$ warstw, gdzie dla $i \in [n+1]$ i-tą warstwę stanowią te wierzchołki, których ciągi binarne mają dokładnie i jedynek (warstwa zawiera zatem wierzchołki oddalone o i od wierzchołka zerowego $(\overline{0})$).

Definicja 1.1.3. *Dla dwóch wierzchołków hiperkostki defininiujemy:* $u\Delta v = \{i : u_i \neq v_i\}$*, gdzie* $(u_0, ...u_{n-1})$ *i* $(v_0, ..., v_{n-1})$ *to ciagi binarne wierzchołków* u *i* v *odpowienio* ($|u\Delta v|$ *wyznacza odległość wierzchołków w hiperkostce).*

Definicja 1.1.4. Numerowaniem klasycznym (naturalnym) *hiperkostki nazwiemy takie numerowanie* $\varphi : V(Q_n) \to \{1, ..., |V(Q_n)|\}$ *jej wierzchołków, że* $\varphi(v) = 1 + \sum_i v_i \cdot 2^i$



Definicja 1.1.5. Numerowaniem warstwowym *hiperkostki nazwiemy jej numerowanie w kolejności przeszukiwania grafu wszerz zaczynając od wierzchołka* $\overline{0}$ *z wybieraniem sąsiadów w kolejności leksykograficznej.*



Uwaga 1. *Jest to takie numerowanie* $\varphi : V(Q_n) \to \{1, ..., |V(Q_n)|\}$ *jej wierzchołków, że wierzchołki z i-tej warstwy otrzymują numery od* $\sum_{j=0}^{i-1} \binom{n}{j} + 1$ *do* $\sum_{j=0}^{i} \binom{n}{j}$. *W obrębie jednej warstwy numery przyznawane są przeciwnie do kolejności leksykograficznej na odwróconych słowach.* $\varphi(v) > \varphi(u) \Leftrightarrow (\sum_{i=0}^{n} v_i > \sum_{i=0}^{n} u_i) \vee ((\sum_{i=0}^{n} v_i = \sum_{i=0}^{n} u_i) \wedge (\sum_{i=0}^{n} 2^{n-i} v_i < \sum_{i=0}^{n} 2^{n-i} u_i))$

Dowód. Indukcyjnie po warstwach.
Dla warstwy 0 oczywiste.
Zakładając, że i-ta warstwa jest ponumerowana w tym porządku weźmy dwa wierzchołki u, v z warstwy $i+1$: $u = (\overline{y_1}, 1, \overline{x}), v = (\overline{y_2}, 0, \overline{x})$.
Jeśli $\overline{y_1}$ zawiera same 0, to $\overline{y_2}$ zawiera dokładnie jedną 1, sąsiedzi tych wierzchołków z poprzedniej warstwy o namniejszych numerach to odpowiednio $(\overline{0}, 0, x), (\overline{0}, 0, x)$, tak więc zostaną ponumerowane jako sąsiedzi tego samego wierzchołka, jednak u otrzyma mniejszy numer jako sąsiad mniejszy leksykograficznie.
Jeśli $\overline{y_1}$ zawiera 1, to $\overline{y_2}$ też, więc sąsiedzi tych wierzchołków z poprzedniej warstwy o namniejszych numerach to odpowiednio $(\overline{y_1'}, 1, \overline{x}), (\overline{y_2'}, 0, \overline{x})$, gdzie $\overline{y_1'}$ i $\overline{y_2'}$, to odpowiednio $\overline{y_1}$ i $\overline{y_2}$ z pierwszymi 1 zamienionymi na 0. Z założenia indukcyjnego sąsiad u ma mniejszy numer niż sąsiad v, więc u ma mniejszy numer niż v. □

Definicja 1.1.6. *Dla grafu* G *oraz wierzchołka* $v \in V(G)$ *definiujemy* sąsiedztwo wierzchołka *jako zbiór wierzchołków połączonych z nim krawędzią:* $N(v) = \{u \in V(G) : (u, v) \in E(G)\}$.

Definicja 1.1.7. *Dla grafu* G *oraz zbioru wierzchołków* $S \subseteq V(G)$ *definiujemy* sąsiedztwo zbioru wierzchołków *jako zbiór tych sąsiadów wierzchołków ze zbioru, które same do tego zbioru nie należą:* $N(S) = (\bigcup_{v \in S} N(v)) \backslash S$

Definicja 1.1.8. *Dla grafu* G *oraz zbioru wierzchołków* $S \subseteq V(G)$ *definiujemy* wnętrze zbioru wierzchołków *jako zbiór tych wierzchołków z* S, *których wszyscy sąsiadzi również należą do tego zbioru:* $In(S) = \{v \in S : N(v) \subseteq S\}$

Definicja 1.1.9. *Dla danego* $V \subseteq V(G)$ $\quad G[V] = (V, \{uv \in E(G) : u, v \in V\})$ *oznacza* podgraf indukowany *przez podzbiór wierzchołków* V.

Definicja 1.1.10. *Dla danego* $V \subseteq V(G)$ $\quad G - V = G[V(G) \setminus V]$ *oznacza* graf G z usuniętymi wierzchołkami V.

Definicja 1.1.11. *Dla danego grafu* G
$G^2 = (V(G), E(G) \cup \{uv : \exists_{w \in V(G)} uw \in E(G) \cap wv \in E(G)\})$ *oznacza* kwadrat grafu, *czyli graf z dodanymi krawędziami między wierzchołkami oddalonymi o co najwyżej 2.*

1.2. Podstawy kombinatoryczne

$\binom{2n}{n} = \frac{2^{2n}\Gamma(n+\frac{1}{2})}{\sqrt{\pi}\Gamma(n+1)}, \qquad \binom{2n+1}{n} = \binom{2n+1}{n+1} = \frac{2^{2n+1}\Gamma(n+\frac{3}{2})}{\sqrt{\pi}\Gamma(n+2)}$

$\Gamma(z) = \int\limits_0^\infty x^{z-1}e^{-x}dx$ dla $n \in \mathbb{N}$ $\Gamma(n) = (n-1)!$,

ogólniej dla $x \in \mathbb{R}, x > 1$ $\frac{\Gamma(x+1)}{\Gamma(x)} = x, \qquad \frac{\Gamma(x+\frac{1}{2})}{\Gamma(x)} < \frac{\Gamma(x+1)}{\Gamma(x+\frac{1}{2})} \Rightarrow \sqrt{x - \frac{1}{2}} < \frac{\Gamma(x+\frac{1}{2})}{\Gamma(x)} < \sqrt{x}$

Daje to ograniczenia: $\frac{2^{2n}}{\sqrt{\pi(n+\frac{1}{2})}} < \binom{2n}{n} < \frac{2^{2n}}{\sqrt{\pi n}}, \qquad \frac{2^{2n+1}}{\sqrt{\pi(n+\frac{3}{2})}} < \binom{2n+1}{n} = \binom{2n+1}{n+1} < \frac{2^{2n+1}}{\sqrt{\pi(n+1)}}$

Lub równoważnie: $\frac{2^n}{\sqrt{\pi(\lceil\frac{n}{2}\rceil+\frac{1}{2})}} < \binom{n}{\lfloor\frac{n}{2}\rfloor} = \binom{n}{\lceil\frac{n}{2}\rceil} < \frac{2^n}{\sqrt{\pi\lceil\frac{n}{2}\rceil}}$

Lemat 2. *Dla* $k \leqslant \lfloor\frac{n+1}{2}\rfloor$ *zachodzi ograniczenie* $\sum_{i=0}^{k-1}\binom{n}{i} \leqslant 2^{n-1}\frac{\binom{n}{k}}{\binom{n}{\lfloor\frac{n}{2}\rfloor}}$

Dowód. (Uogólnienie dowodu z podobnego lematu dla $n = 2m, k < m$ z [1])
Załóżmy najpierw, że $k < \lfloor\frac{n}{2}\rfloor$
Zdefiniujmy $c = \frac{\binom{n}{k}}{\binom{n}{\lfloor\frac{n}{2}\rfloor}} < 1, t = \lfloor\frac{n}{2}\rfloor - k$, $A = \sum_{i=0}^{k-1}\binom{n}{i}$, $B = \sum_{i=k}^{\lfloor\frac{n}{2}\rfloor-1}\binom{n}{i}$

$\forall_{1 \leqslant i \leqslant k}$ $\frac{\binom{n}{k-i}}{\binom{n}{\lfloor\frac{n}{2}\rfloor-i}} < \frac{\binom{n}{k-i+1}}{\binom{n}{\lfloor\frac{n}{2}\rfloor-i+1}} \Leftrightarrow \frac{k-c+1}{n-k+c} < \frac{\lfloor\frac{n}{2}\rfloor-c+1}{\lceil\frac{n}{2}\rceil+c}$,

co wynika z szeregu prostych nierówności $\frac{k-c+1}{n-k+c} \leqslant \frac{k-c+1}{k+c+1} < \frac{\lfloor\frac{n}{2}\rfloor-c+1}{\lfloor\frac{n}{2}\rfloor+c+1} \leqslant \frac{\lfloor\frac{n}{2}\rfloor-c+1}{\lceil\frac{n}{2}\rceil+c}$)

Daje nam to ograniczenia $\forall_{1 \leqslant i \leqslant k}$ $\frac{\binom{n}{k-i}}{\binom{n}{\lfloor\frac{n}{2}\rfloor-i}} < c$.

Suma ostatnich t wyrazów szeregu A jest majoryzowana przez $c \cdot B$, wcześniejszych t przez c razy suma ostatnich t (a więc przez $c^2 \cdot B$). Daje nam to oszacowanie $A < (c + c^2 + c^3 + ... + c^{\lfloor\frac{k}{t}\rfloor}) \cdot B < (c + c^2 + c^3 + ...) \cdot B = \frac{c}{1-c} \cdot B$. Jednocześnie $A + B = \sum_{i=0}^{\lfloor\frac{n}{2}\rfloor-1}\binom{n}{i} < 2^{n-1}$.
$A = c \cdot A + (1-c) \cdot A = c(A + \frac{1-c}{c}A) < c \cdot (A + B) < c \cdot 2^{n-1}$.
Pozostaje udowodnić przypadki większych k:

Dla $n = 2m, k = m$ $\sum_{i=0}^{m-1}\binom{2m}{i} = 2^{2m-1} - \frac{1}{2}\binom{2m}{m} < 2^{2m-1} = 2^{n-1} \cdot \frac{\binom{n}{k}}{\binom{n}{\lfloor\frac{n}{2}\rfloor}}$

Dla $n = 2m+1, k = m$ $\sum_{i=0}^{m-1}\binom{2m+1}{i} = 2^{2m} - \binom{2m+1}{m} < 2^{2m} = 2^{n-1} \cdot \frac{\binom{n}{k}}{\binom{n}{\lfloor\frac{n}{2}\rfloor}}$

Dla $n = 2m+1, k = m+1$ $\sum_{i=0}^{m}\binom{2m+1}{i} = 2^{2m} = 2^{n-1} \cdot \frac{\binom{n}{k}}{\binom{n}{\lfloor\frac{n}{2}\rfloor}}$ (jedyna nieostra nierówność)

□

1.3. Własność ekspansji

Definicja 1.3.1. *Graf* G *posiada własność* ε–ekspansji wierzchołkowej, *jeżeli dla każdego zbioru wierzchołków* $S \subseteq V(G)$ *takiego, że* $|S| \leqslant \frac{|V(G)|}{2}$ *zachodzi* $|N(S)| \geqslant \varepsilon \cdot |S|$

Lemat 3. *Zbiór pierwszych l wierzchołków hiperkostki według numerowania warstwowego posiada maksymalne wnętrze wśród zbiorów wielkości l.*

Jest to jeden z lematów dowodzonych w pracy [2].

Lemat 4. *Dla hiperkostki do udowodnienia własności ε_n–ekspansji wierzchołkowej wystarczy rozważyć zbiory S postaci $S_k, k \leqslant 2^{n-1}$.*

Dowód. Weźmy dowolne $S \subseteq V(G), l = |S| + |N(S)|$ z Lematu 1.3 wynika, że $\frac{|N(S)|}{|S|} = \frac{|N(S)|+|S|}{|S|} - 1 \geqslant \frac{|S_l|}{|In(S_l)|} - 1 = \frac{|S_l \setminus In(S_l)|}{|In(S_l)|} \geqslant \frac{|N(In(S_l))|}{|In(S_l)|}$. Z definicji S_l wynika, że $In(S_l) = S_k$ dla $k = In(S_l)$.
Pozostaje udowodnić, że wystarczy rozważyć te S_k, że $k \leqslant 2^{n-1}$
Dla $l = |N(S)| + |S| \geqslant (\varepsilon_n + 1) \cdot 2^{n-1}$ mamy $|S| > 2^{n-1}$ lub $|N(S)| \geqslant \varepsilon_n |S|$, wystarczy więc rozważyć przypadek $l < (\varepsilon_n + 1) \cdot 2^{n-1}$.
Dla $n = 2m+1$ weźmy $k = 2^{n-1} = \sum_{i=0}^{m} \binom{2m+1}{i}$, wtedy $S_k =$ pełne $m+1$ pierwszych warstw i $N(S_k) =$ warstwa $m+1$. Przykład ten pokazuje, że $\varepsilon_n \leqslant \frac{\binom{2m+1}{m+1}}{2^{2m}}$, więc $l < 2^{2m} + \binom{2m+1}{m+1} \Rightarrow S_l$ mieści się w piewszych $m+2$ warstwach $\Rightarrow S_k = In(S_l)$ mieści się w pierwszych $m+1$ warstwach $\Rightarrow k \leqslant 2^{2m} = 2^{n-1}$.
Dla $n = 2m$ weźmy $k = 2^{n-1} = \sum_{i=0}^{m-1} \binom{2m}{i} + \frac{1}{2}\binom{2m}{m}$, wtedy $S_k =$ pełne m pierwszych warstw + połowa środkowej. W środkowej warstwie pierwsze $\binom{2m-1}{m-1} = \frac{1}{2}\binom{2m}{m}$ wierzchołków to dokładnie te, których ciągi binarne kończą się na 1. Wtedy też $S_k \cup N(S_k)$ to dokłanie pełne $m+1$ pierwszych warstw plus te wierzchołki z warstwy $m+2$, które kończą się na $1 \Rightarrow |N(S_k)| = \binom{2m-1}{m} + \binom{2m-1}{m} = \binom{2m-1}{m-1} + \binom{2m-1}{m} = \binom{2m}{m}$. Przykład ten pokazuje, że $\varepsilon_n \leqslant \frac{\binom{2m}{m}}{2^{2m-1}}$, więc $l < 2^{2m-1} + \binom{2m}{m} \Rightarrow S_l$ mieści się w piewszych $m+1$ warstwach plus tych wierzchołkach z warstwy $m+2$, które kończą się na $1 \Rightarrow k \leqslant 2^{n-1}$. □

Wniosek 5. *Hiperkostka wymiaru n nie posiada własności $\frac{2\sqrt{2}}{\sqrt{\pi n}}$–ekspansji wierzchołkowej.*

Dowód. Dla $n = 2m+1$
$\frac{|N(S_{2^{2m}})|}{|S_{2^{2m}}|} = \frac{\binom{2m+1}{m+1}}{2^{2m}} = \frac{2^{2m+1}}{2^{2m}\sqrt{\pi(m+1)}} = \frac{2}{\sqrt{\pi(m+1)}} = \frac{2}{\sqrt{\pi(\frac{n}{2}+\frac{1}{2})}} = \frac{2\sqrt{2}}{\sqrt{\pi(n+1)}} < \frac{2\sqrt{2}}{\sqrt{\pi n}}$.
Dla $n = 2m$ $\frac{|N(S_{2^{2m-1}})|}{|S_{2^{2m-1}}|} = \frac{\binom{2m}{m}}{2^{2m-1}} < \frac{2^{2m}}{2^{2m-1}\sqrt{\pi(m+1)}} = \frac{2}{\sqrt{\pi m}} = \frac{2}{\sqrt{\pi \cdot \frac{n}{2}}} = \frac{2\sqrt{2}}{\sqrt{\pi n}}$. □

Twierdzenie 6. *Hiperkostka Q_n posiada własność $\frac{1}{\sqrt{\pi n}}$–ekspansji wierzchołkowej.*

Dowód. Jeśli $k = \sum_{i=0}^{r} \binom{n}{i}$ (pełne $r+1 \leqslant \lfloor \frac{n}{2} \rfloor + 1$ warstw), to $\frac{|N(S_k)|}{|S_k|} = \frac{\binom{n}{r+1}}{\sum_{i=0}^{r}\binom{n}{i}} > \frac{\binom{n}{r+1}\binom{n}{\lfloor \frac{n}{2} \rfloor}}{2^{n-1}\binom{n}{r+1}} = \frac{\binom{n}{\lfloor \frac{n}{2} \rfloor}}{2^{n-1}} > \frac{2^n}{2^{n-1}\sqrt{\pi(\lceil \frac{n}{2} \rceil + \frac{1}{2})}} = \frac{2}{\sqrt{\pi(\lceil \frac{n}{2} \rceil + \frac{1}{2})}} \geqslant \frac{2\sqrt{2}}{\sqrt{\pi(n+\frac{3}{2})}} \geqslant \frac{2}{\sqrt{\pi n}}$ (dla $n \geqslant 2$).
($n = 2m+1, r = m$ rozważone w 4)
Jeśli $k = \sum_{i=0}^{r} \binom{n}{i} + \binom{n-1}{r}$ (pełne $r+1 \leqslant \lfloor \frac{n}{2} \rfloor + 1$ warstw ? TODO + te wierzchołki z warstwy $r+2$, których ciągi binarne kończą się na 1).
$|N(S_k) \cup S_k| = \sum_{i=0}^{r+1} \binom{n}{i} + \binom{n-1}{r+1} \Rightarrow |N(S_k)| = 2 \cdot \binom{n-1}{r+1}$
$\frac{|N(S_k)|}{|S_k|} = \frac{2 \cdot \binom{n-1}{r+1}}{\sum_{i=0}^{r}\binom{n}{i} + \binom{n-1}{r}} > \frac{2 \cdot \binom{n-1}{r+1}\binom{n}{\lfloor \frac{n}{2} \rfloor}}{2^{n-1} \cdot \binom{n}{r+1} + \binom{n-1}{r} \cdot \binom{n}{\lfloor \frac{n}{2} \rfloor}} = \frac{2 \cdot \binom{n-1}{r+1}\binom{n}{\lfloor \frac{n}{2} \rfloor}}{2^{n-1} \cdot (\binom{n-1}{r} + \binom{n-1}{r+1}) + \binom{n-1}{r} \cdot \binom{n}{\lfloor \frac{n}{2} \rfloor}} > \frac{2 \cdot \binom{n-1}{r+1}\binom{n}{\lfloor \frac{n}{2} \rfloor}}{2^n \cdot \binom{n-1}{r+1} + \binom{n-1}{r} \cdot \binom{n}{\lfloor \frac{n}{2} \rfloor}} = \left(\frac{2^{n-1}}{\binom{n}{\lfloor \frac{n}{2} \rfloor}} + \frac{\binom{n-1}{r}}{2 \cdot \binom{n-1}{r+1}} \right)^{-1} > \left(\frac{\sqrt{\pi(n+\frac{3}{2})}}{2\sqrt{2}} + \frac{1}{2} \right)^{-1} = \frac{2\sqrt{2}}{\sqrt{\pi(n+\frac{3}{2})}+\sqrt{2}} > \frac{2}{\sqrt{\pi n}}$ (dla $n \geqslant 7$).

W pozostałych przypadkach można otrzymać ograniczenie choć dużo gorsze wiedząc, że dodanie wierzchołka do S zmniejszy $N(S)$ o co najwyżej 1.
Weźmy teraz $\sum_{i=0}^{r}\binom{n}{i} < k < \sum_{i=0}^{r}\binom{n}{i} + \binom{n-1}{r}$
$\frac{|N(S_k)|}{|S_k|} > \frac{\binom{n}{r+1}-\binom{n-1}{r}}{\sum_{i=0}^{r}\binom{n}{i}+\binom{n-1}{r}} = \frac{\binom{n-1}{r+1}}{\sum_{i=0}^{r}\binom{n}{i}+\binom{n-1}{r}} \geqslant \frac{\frac{1}{2}\binom{n}{r+1}}{\sum_{i=0}^{r}\binom{n}{i}+\frac{1}{2}\binom{n}{r+1}} \left(\frac{\sqrt{\pi(n+\frac{3}{2})}}{\sqrt{2}}+1\right)^{-1} > \frac{1}{\sqrt{\pi n}}$ (dla $n \geqslant 7$).
Analogicznie da $\sum_{i=0}^{r}\binom{n}{i} + \binom{n-1}{r} < k < \sum_{i=0}^{r+1}\binom{n}{i}$
$\frac{|N(S_k)|}{|S_k|} > \frac{2\binom{n-1}{r+1}-\binom{n-1}{r+1}}{\sum_{i=0}^{r+1}\binom{n}{i}} = \frac{\binom{n-1}{r+1}}{\sum_{i=0}^{r+1}\binom{n}{i}} > \frac{\binom{n-1}{r+1}}{2^{n-1}\frac{\binom{n}{r+1}}{\binom{n}{\lfloor\frac{n}{2}\rfloor}}+\binom{n}{r+1}} \geqslant \frac{\binom{n}{r+1}\binom{n}{\lfloor\frac{n}{2}\rfloor}}{(2^{n-1}+\binom{n}{\lfloor\frac{n}{2}\rfloor})\binom{n}{r+1}} = \frac{\binom{n}{\lfloor\frac{n}{2}\rfloor}}{2^{n-1}+\binom{n}{\lfloor\frac{n}{2}\rfloor}} >$
$\left(\frac{\sqrt{\pi(n+\frac{3}{2})}}{\sqrt{2}}+1\right)^{-1} > \frac{1}{\sqrt{\pi n}}$ (dla $n \geqslant 7$).
Dla przypadków $n \leqslant 6$ można ręcznie sprawdzić wszystkie 2^{n-1} przypadków, aby również otrzymać oszacowanie $\frac{1}{\sqrt{\pi n}}$. $\square$

Rozdział 2

Problemy na wadliwej hiperkostce

2.1. Graf z wadami

Definicja 2.1.1. *W grafie G możemy wyróżnić niektóre wierzchołki (czasem również krawędzie) i oznaczyć jako wadliwe. Graf z niepustym takim wyróżnionym zbiorem wierzchołków wadliwych $F \subseteq V(G)$ nazywamy* grafem z wadami *(lub* grafem wadliwym*)*

Wadliwe wierzchołki (i/lub krawędzie) najczęściej traktowane są jako usunięte z grafu – mówimy w tym przypadku o grafie $G - F$. Wyróżnianie wadliwych wierzchołków w grafie zamiast definiowania nowego grafu jest umotywowane głównie w przypadkach, gdy pełny graf łatwo zdefiniować i zapisać w pamięci małej względem jego rozmiaru (np. klika, hiperkostka, graf de Bruijna), a zbiór wadliwych wierzchołków jest również mały.

2.2. Spójność wadliwej hiperkostki

Ten podrozdział jest napisany w większości na podstawie [3].

Uwaga 7. *Aby zbadać spójność grafu $G - F$ dla spójnego grafu G wystarczy sprawdzić czy wciąż istnieje ścieżka pomiędzy dowolnymi dwoma wierzchołkami, które oryginalnym grafie sąsiadowały z jakimś spośród usuniętych wierzchołków (wszystkie takie wierzchołki należą do jedenj spójnej składowej).*

Dowód. Aby udowodnić spójność trzeba pokazać, że istnieje ścieżka pomiędzy dowolnymi dwoma wierzchołkami, jednak skoro w oryginalnym grafie taka ścieżka istniała, to w nowym grafie jedyną przeszkodą jest to, że na tej ścieżce mogły występować wierzchołki, które zostały usunięte. Taką scieżkę można naprawić wstawiając w miejsca od pierwszego do ostatniego wystąpienia wierzchołka usuniętego ścieżkę pomiędzy odpowiednimi ich sąsiadami istniejącą w pomniejszonym grafie. □

2.2.1. Podejście ekspansywne

Twierdzenie 8. *Niech graf G posiada własność ε–ekspansji wierzchołkowej z $\varepsilon > 0$ i maksymalny stopień wierzchołka Δ, oraz dana jest wyrocznia zwracająca dla danego wierzchołka listę jego sąsiadów. Wtedy istnieje algorytm, który otrzymuje na wejściu zbiór $F \subseteq V(G)$ oraz ε i testuje spójność $G - F$ w czasie $O\left(\frac{|F|^2 \cdot \Delta^2 \cdot \log(|V(G)|)}{\varepsilon}\right)$*

Lemat 9. *Spójna składowa $S \subseteq V(G) \backslash F$ grafu $G - F$ jest jednego z dwóch typów:*

- *główna* – $|S| > \frac{|V(G)|}{2}$

- *mała* – $|S| \leqslant \frac{|F|}{\varepsilon}$

Uwaga 10. *Co prawda dla dużego $|F|$ i małego ε może być tak, że składowa jest jednocześnie główna i mała, jednak po pierwsze jest to przypadek mało interesujacy, gdyż wtedy zwykłe przeszukiwanie grafu spełnia tezę twierdzenia, a po drugie przypadek ten nie psuje w żaden sposób otrzymywanego algorytmu. W lemacie istotne jest to, że w grafie nie ma składowych średnich wielkości.*

Fakt 11. *Może być tylko jedna składowa główna.*

Dowód. (Lematu)
Weźmy spójną składową S grafu $G - F$ ($N_{G-F}(S) = 0$), jeżeli $S \leqslant \frac{|V(G)|}{2}$, to z własności ε–ekspansji wierzchołkowej grafu G $|N_G(S)| \geqslant \varepsilon \cdot |S|$ (gdzie S jest teraz traktowane jako podzbiór wierzchołków grafu G). Gdyby zachodziło $|S| > \frac{|F|}{\varepsilon}$, to mielibyśmy $|N_G(S)| > \frac{\varepsilon \cdot |F|}{\varepsilon} = |F|$, co daje sprzeczność ponieważ aby w grafie $G - F$ to sąsiedztwo było puste z grafu G trzeba usunąć co najmniej $N_G(S)$ wierzchołków. □

Dowód. (Twierdzenia)
Chcemy sprawdzić, czy wszyscy sąsiedzi wierzchołków usuniętych należą do tej samej spójnej składowej. Na podstawie lematu 9, jeśli składowa zawierająca taki wierzchołek jest większa niż $\frac{|F|}{\varepsilon}$, to jest to składowa główna. Jeżeli wszystkie takie wierzchołki spełniają ten warunek, to $G - F$ jest spójny. Jeżeli natomiast, któraś z tych składowych okaże się mała, to $G - F$ nie jest spójny.
Wystarczy więc uruchomić liniowe przeszukiwanie grafowe w każdym wierzchołku sąsiadującym z wierzchołkiem wadliwym i przerywać po przejrzeniu $\frac{|F|}{\varepsilon}$ wierzchołków.
Algorytm liniowego przeszukiwania grafowego uruchamiany jest co najwyżej $|F| \cdot \Delta$ razy. Za każdym razem przeglądamy co najwyżej $\frac{|F|}{\varepsilon}$ wierzchołków. Dla każdego przeglądanego wierzchołka sprawdzamy co najwyżej Δ sąsiadów, a odpowiedź wyroczni zajmuje $O(log(|V(G)|))$ czasu. Daje to złożoność z tezy twierdzenia. □

Wniosek 12. *Ponieważ zgodnie z twierdzeniem 6 hiperkostka Q_n posiada własność $\frac{1}{\sqrt{\pi n}}$–ekspansji wierzchołkowej, oraz można znaleźć wszystkich sąsiadów wierzchołka w czasie liniowym od ich ilości powyższy algorytm testuje spójność wadliwej hiperkostki w czasie $O(|F|^2 \cdot n^{3.5})$ (wyrażonego w ilości operacji arytmetycznych).*

Uwaga 13. *Ze względu na długość zapisu identyfikatora wierzchołka liniową od wymiaru hiperkostki nie da się przeprowadzać operacji na wierzchołkach w czasie szybszym niż n. To dolne ograniczenie jest osiągalne przy przechowywaniu przejrzanych wierzchołków w hashmapie (czas oczekiwany operacji $O(n)$, złożoność pamięciowa całej struktury $O(n^{0.5}|F|)$), lub w drzewie prefiksowym (czas pesymistyczny operacji $O(n)$, złożoność pamięciowa całej struktury $O(n^{1.5}|F|)$). Pozwala to w łatwy sposób uzyskać efektywną wyrocznię, a więc i algorytm o złożoności z wniosku.*

pseudokod i uwagi

W algorytmie wykorzystywana jest sturktura T z operacjami

- $Insert(v, T)$ wstawiającą wierzchołek v do struktury T

- $Retrieve(v, T)$ zwracająca binarną informacje o obecności wierzchołka v w strukturze T

które wymagają $O(n)$ czasu na wykonanie (jak w uwadze 13).
Przeszukiwanie grafowe odbywa się przy pomocy funkcji o pseudokodzie:

```
DFS(v){
    counter++;
    Insert(v,T);
    if(counter ⩾ size)   return(TRUE);
    foreach(u ∈ N(v)){
        if(Retrieve(u,T) == FALSE){
            if(DFS(u))   return(TRUE);
        }
    }
    return(FALSE);
}
```

Spójność sprawdzana jest przy pomocy funkcji głównej o pseudokodzie:

```
Conectivity(n,F){
    T2 = empty_structureT();
    counter = 0;
    foreach(f ∈ F){
        Insert(f,T2);
        counter++;
    }
    size = sqrt(π * n) * counter;
    foreach(f ∈ F){
        foreach(v ∈ N(f)){
            if(Retrieve(v,T2) == FALSE){
                counter = 0;
                T = T2;
                if(DFS(v) == FALSE)   return(FALSE);
            }
        }
    }
    return(TRUE);
}
```

Uwaga 14. *Aby przeiterować po $N(v)$ wystarczy przeiterować się po współrzędnych uzyskując sąsiada poprzez zanegowanie tej współrzędnej w zapisie bitowym v.*

Uwaga 15. *Można użyć dodatkowej struktury T w której przechowywane są wszystkie wierzchołki z poprzednich wywołań $DFS(v)$ z funkcji głównej. Wtedy przy kolejnych użyciach $DFS(v)$ można sprawdzać, czy wierzchołek nie był już wcześniej w jakiejś składowej (można wtedy od razu zwrócić $TRUE$). Teoretycznie może to zwiększyć złożoność dwukrotnie, jednak w praktyce będzie to dużo szybsze (już nawet z tego wzgledu, że albo inni sąsiedzi tego samego f są oddaleni o 2, albo na drodze staje inny wierzchołek z F), w szczególności przy użyciu bardziej wyszukanych kolejności przeszukiwania (np. próba dojścia do wierzchołka $\overline{0}$).*

2.2.2. Redukcja przy pomocy transformacji ścieżek

Algorytm przedstawiony w poprzednim podrozdziale jest dowodem na to, że testowanie spójności wadliwej hiperkostki może być zrobione wielomianowo ze względu na ilość wad i wymiar hiperkostki. Algorytm ten wykorzystuje jednak bardzo płytko potencjał tak regularnego grafu. W tym podrozdziale przedstawię algorytm, który dzięki głębszemu wykorzystaniu własności hiperkostki otrzymuje lepsze rezultaty złożonościowe.

Definicja 2.2.1. *Na potrzeby tego podrozdziału definiuję ze pracą [3] dla* $F \subseteq V(Q_n)$ *podgraf* $G(F) = (A \cup B \cup F, E)$ *grafu* Q_n*, gdzie* $A = N(F), \quad B = N(A) \backslash F,$
$E = \{uv \in E(Q_n) : u \in A \cup F\}$ *(podgraf zawierający wierzchołki w odległości* $\leqslant 2$ *od wierzchołków wadliwych, plus krawędzie w których jeden z końców jest wadliwy lub z takim sąsiaduje).*

Twierdzenie 16. *Dla* $F \subseteq V(Q_n)$ *graf* $Q_n - F$ *jest spójny wtedy i tylko wtedy gdy dla każdej* C *– spójnej składowej* $Q_n^2[F]$ *spójny jest graf* $G(C) - C$*.*

Transformacje ścieżek w hiperkostce

Definicja 2.2.2. *Dla ścieżki* $W = (v_0, v_1, ..., v_n)$ *(z możliwymi powtórzeniami) w hiperkostce* sekwencją tranzycji *nazywamy ciąg* $\tau = (d_1, d_2, ..., d_n)$*, gdzie* d_i *jest współrzędną na której różnią się ciągi binarne wierzchołków* v_{i-1} *i* v_i*.*

Fakt 17. τ *jest sekwencją tranzycji pewnej uv–ścieżki w* Q_n *wtedy i tylko gdy*
$u \Delta v = \{i \in [n] : \#(\tau, i) \text{ nieparzyste}\}$*, gdzie* $\#(\tau, i)$ *to ilość wystąpień* i *w sekwencji* τ*.*

Dla τ – sekwencji tranzycji uv–ścieżki W definiujemy trzy operacje:

- $swap(\tau_1, i, j, \tau_2) = (\tau_1, j, i, \tau_2)$ dla $\tau = (\tau_1, i, j, \tau_2)$
- $insert_i(\tau_1, \tau_2) = (\tau_1, i, i, \tau_2)$ dla $\tau = (\tau_1, \tau_2), i \in [n]$
- $delete(\tau_1, i, i, \tau_2) = (\tau_1, \tau_2)$ dla $\tau = (\tau_1, i, i, \tau_2)$

Wszystkie te operacje nie zmieniają parzystości wystąpień współrzędnych, dlatego też dowolnie w ten sposób zmodyfikowana sekwencja wciąż jest sekwencją tranzycji pewnej uv–ścieżki w Q_n.

Definicja 2.2.3. *Dwie ścieżki, których sekwencje tranzycji* τ, ρ *spełniają* $\forall_{i \in [n]} \#(\tau, i) = \#(\rho, i)$ *nazywamy* równoważnymi.

Uwaga 18. *Dla dowolnych dwóch uv–ścieżek w* Q_n *istnieje sekwencja operacji* $swap, insert, delete$ *(w tej kolejności bez przeplotów), która przemiania sekwencję tranzycji pierwszej w sekwencję tranzycji drugiej.*

Dowód. Jeśli dwie ścieżki są równoważne, to można jedną przekształcić w drugą przy pomocy samych operacji *swap*.
W przypadku gdy sekwencje mają różne liczności wystąpień współrzędnych, to można je doprowadzić do takich τ', ρ', że $\forall_{i \in [n]} \#(\tau', i) = \#(\rho', i)$ przy pomocy samych operacji *insert* (używanych w dowolnie wybranych wierzchołkach). □

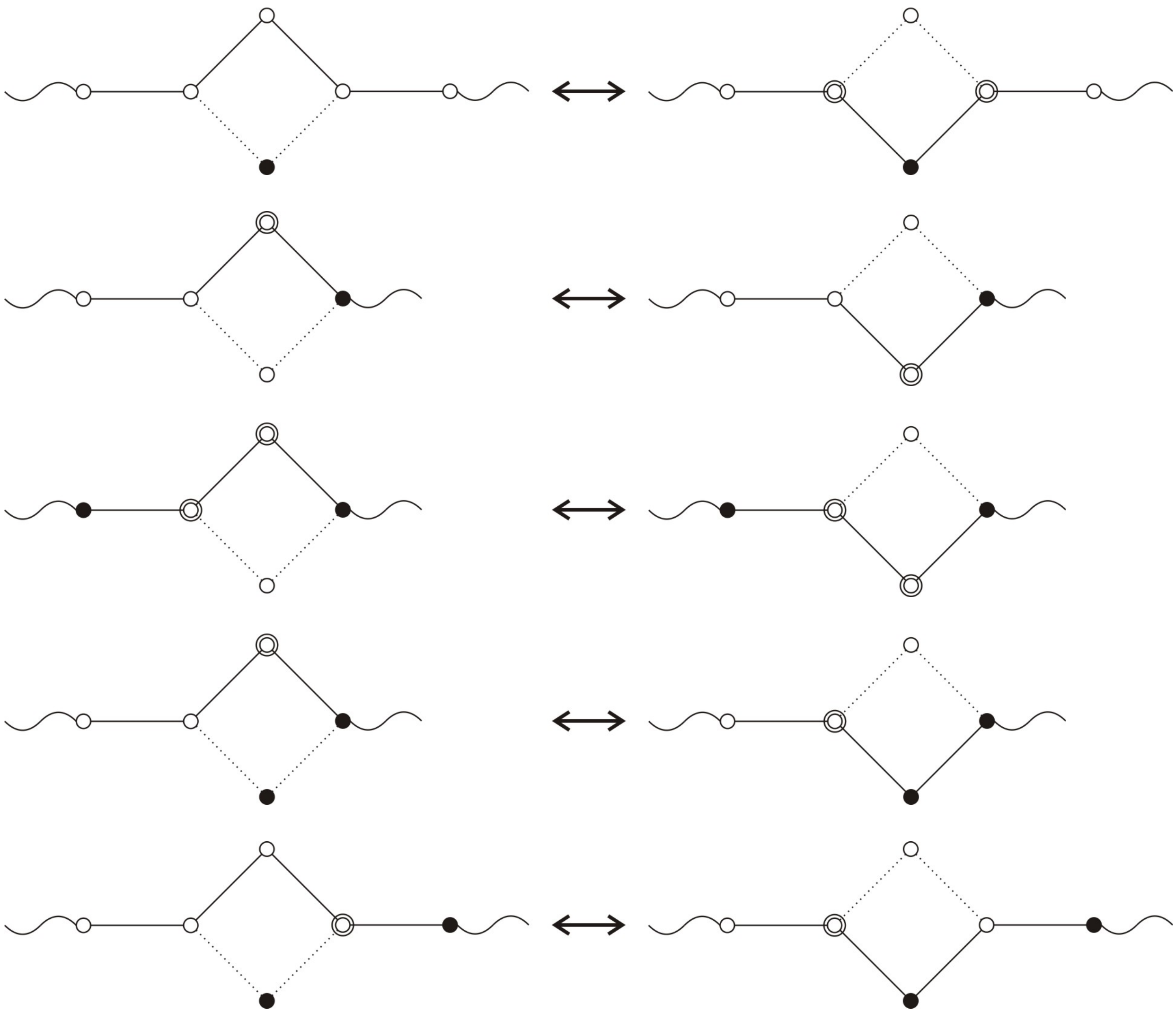
Definicja 2.2.4. *Na potrzeby dowodu twierdzenia 16 dla uv–ścieżki* $W = (w_0, w_1, ..., w_k)$ *(gdzie* $w_0 = u, w_k = v$*) wierzchołek* w_i *nazywamy portem, jeśli nie jest wierzchołkiem wadliwym, ale dokładnie jeden z jego sąsiadów w ścieżce należy do* F *(port musi więc należeć do* A*).*

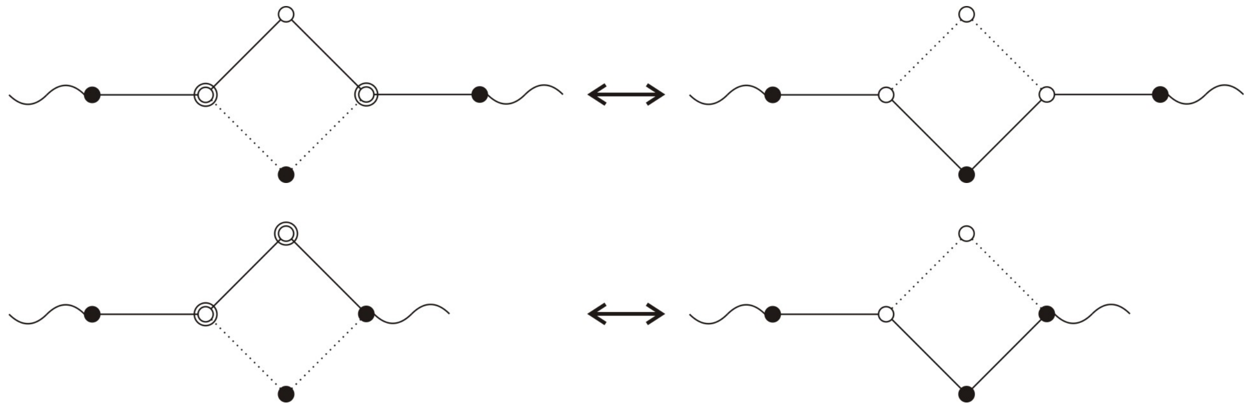
W przypadku tej definicji należy rozróżnić przeplatające się pojęcia wierzchołka grafu i jego wystąpienia na ścieżce – portem nazywane jest konkretne wystąpienie na ścieżce, inne jego wystąpienia nie muszą być portami.
Dla C spójnej składowej $G(F)-F$ przez $p(C, W)$ oznaczamy ilość portów w części W należącej do C.

Lemat 19. *Operacja swap zachowuje parzystość $p(C, W)$.*

Dowód. Dowód stanowi rysunkowe rozpatrzenie wszystkich możliwych przypadków w których w wyniku operacji *swap* powstaje i/lub znika pewien port (przypadki przy końcach ścieżki można "dopełnić" zwykłymi wierzchołkami do przypadków ze środka ponieważ wierzchołki końcowe nie są wadliwe).

○ – wierzchołek zwykły
——— – krawędź ścieżki
● – wierzchołek wadliwy
.... – krawędź spoza ścieżki
◎ – port
∼ – nieistotna reszta ścieżki





Wszystkie rozrysowane tu wierzchołki należą do $G(F)$, a rozrysowane części niewadliwe tworzą podgraf spójny, dlatego też wszystkie te zmiany odbywają się w jednej spójnej składowej $G(F) - F$. Za każdym razem ilość portów zmienia się o 0 lub 2, a więc parzystość pozostaje bez zmian. □

Dowód twierdzenia o lokalnej spójności

Lemat 20. *Niech $F \subseteq V(Q_n)$ takie, że $G(F)$ jest spójne. Jeśli $Q_n - F$ jest spójne, to $G(F) - F$ również.*

Dowód. Załóżmy przeciwnie – istnieją wierzchołki $u, v \in A \cup B = V(G(F) - F)$ dla których istnieje ścieżka P w $Q_n - F$, ale nie ma takiej w $G(F) - F$. Skoro w $G(F) - F$ nie ma takiej ściezki, to w P musi występować wierzchołek x spoza $A \cup B$.
$G(F)$ jest spójne, więc musi istnieć też druga ścieżka R łacząca u z v w tym właśnie grafie, na której występuje wierzchołek $y \in F$.
Podobnie jak w dowodzie lematu 18 ścieżki te mogą być napompowane w wierzchokach x i y odpowidnio sekwencjami operacji *insert*.
Ponieważ x jest oddalone od F dodane do ścieżki P wierzchołki nie uczynią z zadnego wystąpienia x portu i same również nie staną się portami. Ponieważ y należy do F dodane do ścieżki R wierzchołki będą miały dokładnie dwóch sąsiadów z F (tego samego dwukrotnie), a więc nie będą portami.
Oznacza to, że uzyskane ścieżki równoważne mają tyle samo portów we wszystkich spójnych składowych $G(F) - F$ co odpowiadające im nieprzekształcone a na podstawie lematów 18 i 19 ich parzystości między sobą się zgadzają (czyli zgadzają się dla P i R). Na ścieżce P nie ma żadnych portów ponieważ nie występuje w niej żaden wierzchołek wadliwy, daje to sprzeczność ponieważ dla spójnych składowych $G(F) - F$ w której występują u i v ścieżka R ma nieparzyste ilości portów (np. można dobrać taką ścieżkę, która wchodzi do/opuszcza składowe co najwyżej raz). □

Wniosek 21. *Dla $F \subseteq V(Q_n)$ takiego, że $Q_n^2[F]$ jest spójne ze spójności $Q_n - F$ wynika spójność $G(F) - F$.*

Dowód. Jeśli dwa wierzchołki $Q_n^2[F]$ są połączone, to w oryginalnym grafie musiały być w odległości $\leqslant 2$, a więc w $G(F)$ muszą być połączone albo bezpośrednio albo poprzez pojedynczy wierzchołek z A, a więc graf $G(F)$ jest spójny co sprawia, że spełnione są założenia lematu. □

Lemat 22. *Niech $F \subseteq V(Q_n)$ taki, że $G(C) - C$ jest spójne dla kazdej C spójnej składowej $Q_n^2[F]$. Wtedy $Q_n - F$ również jest spójny.*

Dowód. Dla dowolnie wybranych dwóch wierzchołków $u, v \in V(Q_n - F)$ weźmy W – ścieżkę między nimi w pełnym Q_n. Jeśli W nie zawiera wadliwego wierzchołka, to jest poprawną ścieżką w $Q_n - F$. W przeciwnym przypadku znajdujemy na tej ścieżce pierwsze wystąpienie wierzchołka wadliwego. Poprzedni wierzchołek na ścieżce oraz pierwszy kolejny z poza zbioru F są dwoma niewadliwymi wierzchołkami należącymi do $G(C)$, gdzie C jest spójną składową $Q_n^2[F]$ (oddalone o 1 od wadliwych wierzchołków, które są połączone ścieżką samych wadliwych wierzchołków). W $G(C)$ nie ma wadliwych wierzchołków spoza C, ponieważ oznaczałoby to, że taki wierzchołek jest oddalony o $\leqslant 2$ od pewnego wierzchołka z C, a więc byłby z nim połączony w Q_n^2, dlatego też ścieżka ze spójnego z założenia $G(C) - C$ nie zawiera wadliwego wierzchołka. Wystarczy więc wadliwą część ścieżki W zastąpić odpowiednią ścieżką z $G(C) - C$ aby otrzymać poprawną ścieżkę w $Q_n - F$. □

Dowód. (Twierdzenia 16)
Lemat 22 jest implikacją w jedną stronę. W drugą stronę dla spójnego $Q_n^2[F]$ jest dana wnioskiem 21. Wystarczy udowodnić, że nic nie psuje się w przypadku gdy $Q_n^2[F]$ ma więcej niż jedną spójną składową. Dla C – spójnej składowej $Q_n^2[F]$ jeśli $Q_n - F$ jest spójne, to jest takie również $Q_n - C$ (dla wierzchołków spoza $Q_n - F$ te same ścieżki są dobre, dla tych z $F \backslash C$ dowolny sąsiad nalezy do $Q_n - F$, więc również łatwo zbudować ścieżkę), a więc spójne jest również $G(C) - C$ co kończy dowód. □

Algorytm

Stosując powyższe twierdzenie można uzyskać wielomianowy algorytm używając jedynie przeszukiwania grafowego podobnie jak w podrozdziale 2.2.1. Można jednak uzyskać lepsze rezultaty używając dodatkowo struktury *Find–Union* i sprawdzając spójności już w trakcie budowania podgrafów $G(C) - C$.
W algorytmie używane są:

- struktura *Find–Union* D z operacjami:
  - $Make(v, D)$ tworzącą singleton $\{v\}$
  - $Find(v, D)$ zwracającą wskaźnik na zbiór zawierający v
  - $Union(u, v, D)$ łączącą zbior zawierający u ze zbiorem zawierającym v

  których zamortyzowany czas można ograniczyć przez $O(logm)$ (a da się nawet uzyskać $O(log^*m)$), gdzie m jest ilością użyć operacji $Make(v, D)$. Dodatkowo struktura zapewnia możliwość sprawdzenia, czy zawiera więcej niż jeden zbiór (wystarczy pojedynczy licznik inkrementowany przy $Make(v, D)$ i dekrementowany przy $Union(u, v, D)$).

- strukturę T do przechowywania informacji o niektórych wierzchołkach jak binarne drzewo prefiksowe lub hashmapa, przechowywującą dla wierzchołka v_T informacje:
  - wskaźnik do wierzchołka v w strukturze D
  - informacje o wadliwości/braku wadliwości wierzchołka
  - binarną informacje o tym czy wierzchołek był odwiedzony i należy do $F \cup N(F)$

  wspierającą operacje:
  - $Insert(v, T)$ wstawiającą wierzchołek v do struktury T i zwracającą wskaźnik na v_T
  - $Retrieve(v, T)$ zwracającą v_T lub $NULL$ w przypadku gdy v nie ma w strukturze

które wymagają $O(n)$ czasu na wykonanie.

Definiuję pomocniczą funkcję uzyskiwania wierzchołków ze struktury T i inicjalizowania w razie nieobecności:

```
Retrieve2(v, T){
    v_T = Retrieve(v, T);
    if(v_T == NULL){
        v_T = Insert(v, T);
        v_T.healthy = TRUE;
        v_T.visited = FALSE;
        Make(v, D);
    }
    return(v_T);
}
```

Najistotniejszą częścią algorytmu jest procedura (czasami dla wielu wierzchołków z F) $DFS(f)$ znajdująca spójne składowe $G(C) - C$, dla C spójnej składowej $Q_n^2[F]$ zawierającej wadliwy wierzchołek f, o następującym pseudokodzie:

```
DFS(f){
    foreach(u ∈ N(f)){
        u_T = Retrieve2(u, T);
        if(u_T.visited == FALSE){
            u_T.visited = TRUE;
            if(u_T.healthy){
                foreach(v ∈ N(u)){
                    v_T = Retrieve2(v, T);
                    if(v_T.healthy){
                        if(Find(u, D) ≠ Find(v, D))   \\ krawędź uv należy do G(C) − C
                    }else if(v_T.visited == FALSE){
                        v_T.visited = TRUE;
                        DFS(v);   \\ wadliwy wierzchołek należący do C
                    }
                }
            }else   DFS(u);   \\ wadliwy wierzchołek należący do C
        }
    }
}
```

Powyższa prodedura uruchamiana jest z funkcji głównej:

```
Conectivity(n, F){
    T = empty_structureT();
    foreach(f ∈ F){
        f_T = Insert(f, T);
        f_T.healthy = FALSE;
        f_T.visited = FALSE;
    }
```

foreach($f \in F$){
$f_T = Retrieve(f, T)$;
if($f_T.visited == FALSE$){
$f_T.visited = TRUE$;
$D = empty\_structureD()$;
$DFS(f)$;
if($D.counter > 1$) $return(FALSE)$;
}
}
return($TRUE$);
}

Analiza złożoności

Wniosek 23. *Algorytm ma pesymistyczną złożoność czasową i pamięciową* $O(|F| \cdot n^3)$.

Dowód. Dla kazdego wierzchołka z F każdy sąsiad jest przeglądany po jeden raz, dla każdego wierzchołka należącego do $N(F)$ również przeglądani są wszyscy sąsiedzi po razie. Przeglądnięcie jedengo wierzchołka (znalezienie odpowiedniego wierzchołka w T i D) zajmuje $O(n)$, ustawienie właściwości w D zajmuje stały czas po posiadaniu dowiązania do odpowiedniego wierzchołka – daje to złożoność tej części $O(|F| \cdot n^3)$.
Operacja $Make(v, D)$ używana jest dla każdego wierzchołka z $G(C) - C$ po razie dla kazdego C (moze być użyta więcej niż raz dla wierzchołków oddalonych o 2 od F i występujących w różnych $G(C)$). W przypadku $Find(v, D)$ i $Union(u, v, D)$ uruchamiane są one maksymalnie odpowiednio dwa i jeden raz dla kazdego z sąsiadów wierzchołków $N(F)$ – daje to złożoność $O(|F| \cdot n^2 \log(n))$.
Preprocessing i Postprocessing (tworzenie i usuwanie struktur T i D) może być zrobione w czasie liniowym od ich wielkości (w przypadku D można trzymać dowiązanie do wszystkich nieuporządkowanych wierzchołków dodatkowo na liście). W przypadku struktury T wielkość tą można ograniczyć przez $O(|F| \cdot n^3)$ przy użyciu drzewa prefiksowego (lub $O(|F| \cdot n^2)$ przy użyciu hashmapy, która nie pozwala jednak uzyskać odpowiedniej złożoności przy pesmistycznym scenariuszu), zaś w przypadku struktur D łącznie $O(|F| \cdot n^2)$. □

Bibliografia

[1] L. Lovasz, J. Pelikan and K. Vesztergombi. "Discrete Mathematics, Elementary and Beyond." *Undergraduate Texts in Mathematics. New York: Springer, first edition, 2003*

[2] L. H. HARPER, "Optimal Numberings and Isoperimetric Problems on Graphs" *JOURNAL OF COMBINATORIAL THEORY 1, 385-393 (1966)*

[3] Tomas Dvorak, Jirı Fink, Petr Gregor, Vaclav Koubek and Tomasz Radzik, "Efficient connectivity testing of hypercubic networks with faults"

[4] Jirı Fink and Petr Gregor, "Long paths and cycles in hypercubes with faulty vertices"

[5] Jirı Fink and Petr Gregor, "Long pairs of paths in faulty hypercubes"

[6] Frank Harary, John P. Hayes and Horng–Jyh Wu, "A survey of theory of hypercube graphs" *Comput. Math. Applic. Vol. 15, No 4, pp. 277-289, 1988*

[7] Frank Harary, Marilynn Livingston, "Independent domination in hypercubes" *Appl. Math. Lett. Vol. 6, No 3, pp. 27-28, 1993*

[8] Wojciech Rytter & Bartosz Szreder, "Wprowadzenie do kombinatoryki algorytmicznej"

[9] DONALD E. KNUTH, "Generating All Tuples and Permutations" *THE ART OF COMPUTER PROGRAMMING VOLUME 4, FASCICLE 2*

[10] FRANK RUSKEY, "Combinatorial Generation" *Working Version, October 1, 2003*

[11] Tibor Szabo, Emo Weltz, "Unique Sink Orientations of Cubes"

[12] Chi Him Wong, "Novel universal cycle constructions for a variety of combinatorial objects" *Guelph, Ontario, Canada, April, 2015*