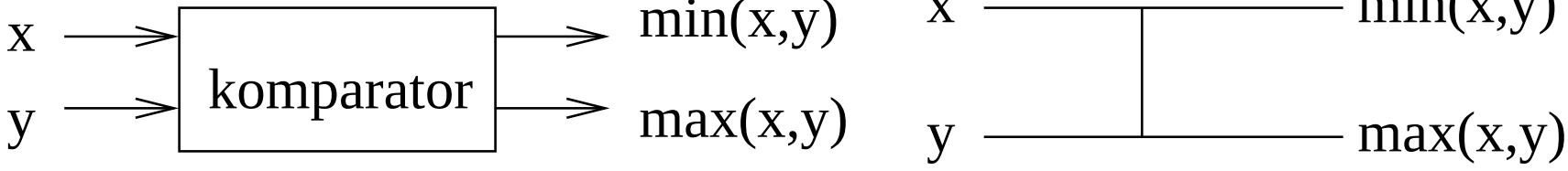
Komparator

Komparator – urządzenie o dwóch wejściach i dwóch wyjściach, które na górnym wyjściu wylicza minimum a na dolnym maksimum z wartości podanych na wejściach.



Komparator działa w czasie $O(1)$.

Sieci komparatorów

Sieć komparatorów - acykliczny graf skierowany:

- wierzchołki:
  - wejścia sieci (z jedną krawędzią wychodzącą)
  - wyjścia sieci (z jedną krawędzią wchodzącą)
  - komparatory (dwie krawędzie wchodzące i dwie wychodzące)
- rodzaje krawędzi:
  - od wejścia sieci do wejścia komparatora (*krawędź wejściowa*)
  - od wyjścia komparatora do wejścia (innego) komparatora
  - od wyjścia komparatora do wyjścia sieci
  - od wejścia do wyjścia sieci (na ogół nie występują)

Sieci komparatorów

Każda sieć ma tyle samo wejść co wyjść. (Suma stopni wejściowych wszystkich wierzchołków musi być równa sumie stopni wyjściowych.)

Rozmiar danych wejściowych – liczba wejść.

Głębokość krawędzi wejściowej: 0.

Głębokość krawędzi wychodzącej z komparatora c: $\max\{d_1, d_2\} + 1$, gdzie d_1, d_2 – głębokości krawędzi wchodzących do komparatora c.

(Definicja głębokości – poprawna, bo graf – acykliczny)

Głębokość komparatora – głębokość jego krawędzi wychodzących.

Głębokość sieci (czas działania) – największa głębokość komparatora w sieci.

(Do chwili t zadziałają wszystkie komparatory głębokości $\leq t$.)

Sieci komparatorów

W n-wejściowej sieci wejścia i wyjścia są ponumerowane od 1 do n.
Jeśli na początku na i-tym wejściu jest wartość a_i, a po zadziałaniu sieci na i-tym wyjściu jest wyznaczona wartość b_i, to mówimy, że sieć dla *ciągu wejściowego* $< a_1, \ldots, a_n >$ wyznacza *ciąg wyjściowy* $< b_1, \ldots, b_n >$.

Sieć standardowa

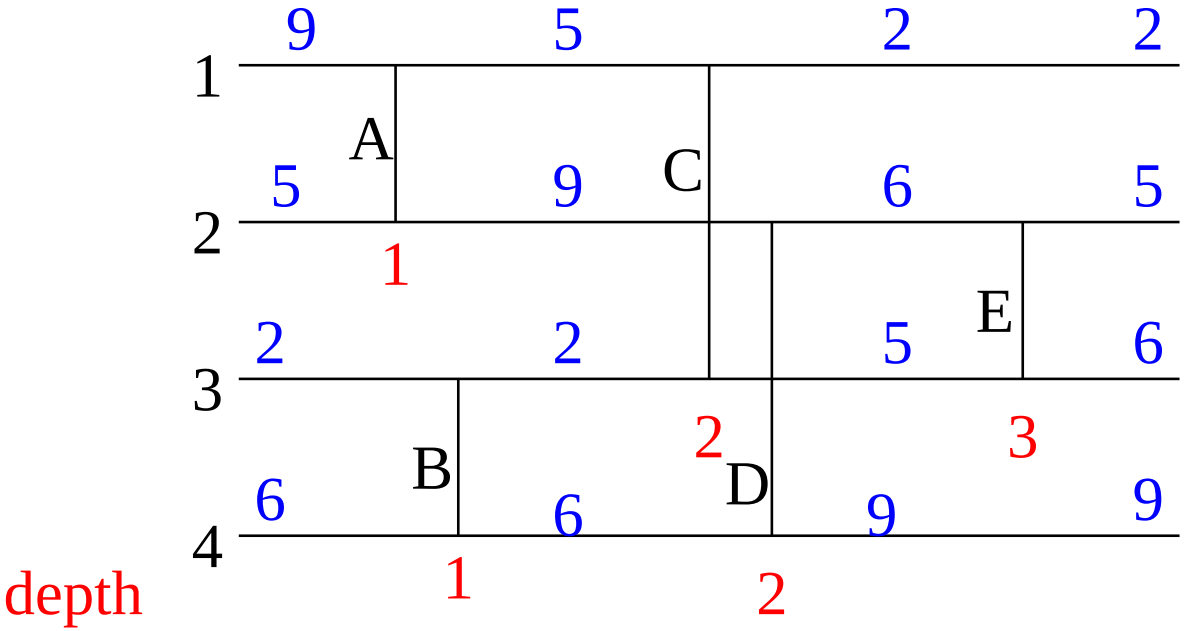
Poetykietujmy krawędzie następująco:

- krawędź wychodząca z wejścia i ma etykietę i;
- krawędź wychodząca z górnego wyjścia komparatora c ma etykietę $\min\{i, j\}$, gdzie i, j – etykiety krawędzi wchodzących do c;
- krawędź wychodząca z dolnego wyjścia komparatora c ma etykietę $\max\{i, j\}$, gdzie i, j – etykiety krawędzi wchodzących do c.

Sieć jest *standardowa*, jeśli przy powyższym etykietowaniu krawędzi: dla każdego i etykieta krawędzi wchodzcej do i-tego wyjścia jest i.

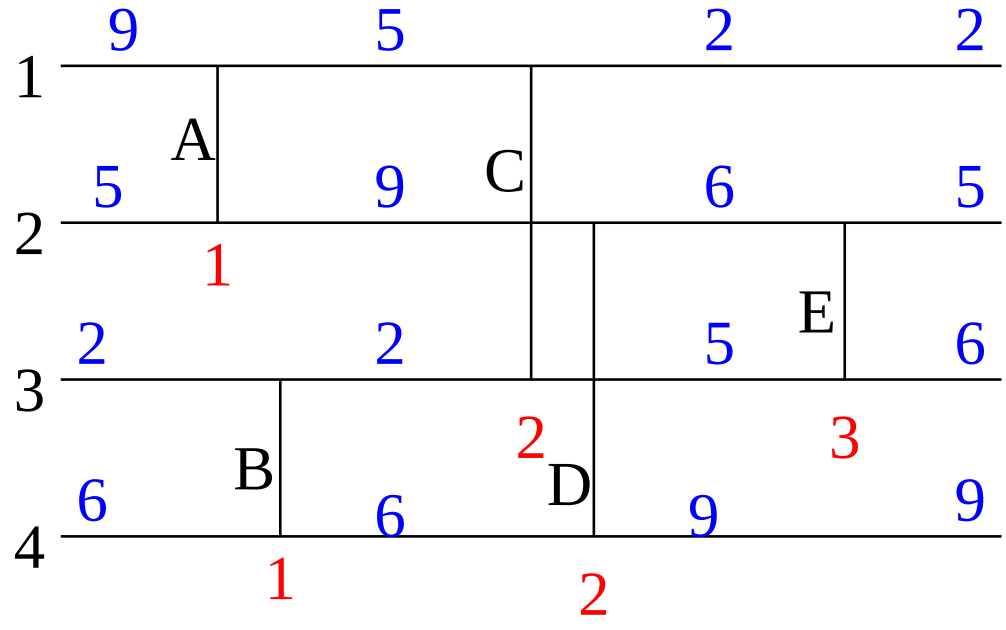
Reprezentacja graficzna

Sieć jest *standardowa*, jeśli można ją przedstawić graficznie w postaci poziomych linii połączonych komparatorami, gdzie każdy komparator jest skierowany w dół.(Na dolnym wyjściu umieszcza maximum.)
i-ta linia od góry – ścieżka krawędzi o etykiecie i od wejścia i do wyjścia i.



Sieci sortujące

Sieć sortująca – sieć komparatorów, która dowolny ciąg wejściowy przekształca na posortowany.



Powyższa sieć jest sortująca:
Po kroku 1 minimalna wartość jest na linii 1 lub 3, a maksymalna na linii 2 lub 4. Po kroku 2 minimalna wartość jest na linii 1, maksymalna na linii 4. Po kroku 3 pozostałe dwie wartości są we właściwym porządku na liniach 2 i 3.

Zasada zero-jedynkowa

Lemat 1. Jeśli sieć komparatorów α dla ciągu wejściowego $a =< a_1, \ldots, a_n >$ wyznacza ciąg wyjściowy $b =< b_1, \ldots, b_n >$, to dla dowolnej funkcji niemalejącej f, α wyznacza dla ciągu wejściowego $f(a) =< f(a_1), \ldots, f(a_n) >$ ciąg wyjściowy $f(b) =< f(b_1), \ldots, f(b_n) >$.

D-d. *Uwaga*: Jeśli na wejściach komparatora są wartości $f(x)$ i $f(y)$, to na górnym wyjściu pojawi się $\min\{f(x), f(y)\}$ a na dolnym $\max\{f(x), f(y)\}$. Ponieważ f – niemalejąca $\min\{f(x), f(y)\} = f(\min\{x, y\})$, oraz $\max\{f(x), f(y)\} = f(\max\{x, y\})$.

Pokażemy, że na każdej krawędzi, na której przy wejściu a pojawia się a_i, przy wejściu $f(a)$ pojawi się $f(a_i)$.

...

Zasada zero-jedynkowa

Indukcja wzgl. głęokości krawędzi:
Na każdej krawędzi o głębokości 0, na której było a_i, pojawi się wartość $f(a_i)$.
krok indukcyjny: Krawędzie o głębokości $d \geq 1$ wychodzą z komparatorów o głębokości d. Krawędzie wejściowe komparatora c o głębokości d mają głębokości $< d$. Niech a_i i a_j – wartości, które pjawią sie na wejściach c przy wejściu a. Z zał. ind. przy wejściu $f(a)$ pojawią się na nich $f(a_i)$ i $f(a_j)$. Przy wejściu a na górnym i dolnym wyjściu c pojawią się odpowiednio $\min\{a_i, a_j\}$ oraz $\max\{a_i, a_j\}$. Z *Uwagi* wynika, że przy wejściu $f(a)$ na górnym i dolnym wyjściu c pojawią się odpowiednio $f(\min\{a_i, a_j\})$ oraz $f(\max\{a_i, a_j\})$.
□

Zasada zero-jedynkowa

Tw. (Zasada zero-jedynkowa) Jeśli sieć o n wejściach poprawnie sortuje wszystkie 2^n ciągi wejściowe zer i jedynek, to poprawnie sortuje wszystkie ciągi wejściowe.

D-d. Załóżmy, że sieć sortuje wszystkie ciągi zero-jedynkowe, ale istnieje ciąg $a =< a_0, \ldots, a_n >$, którego nie sortuje. Tzn. istnieją $a_i < a_j$ takie, że a_i w ciągu wyjściowym pojawia się po a_j. Zdefiniujmy niemalejącą f-cję f:

$$f(x) = \begin{cases} 0, \text{jeśli } x \leq a_i \\ 1, \text{jeśli } x > a_i \end{cases}$$

...

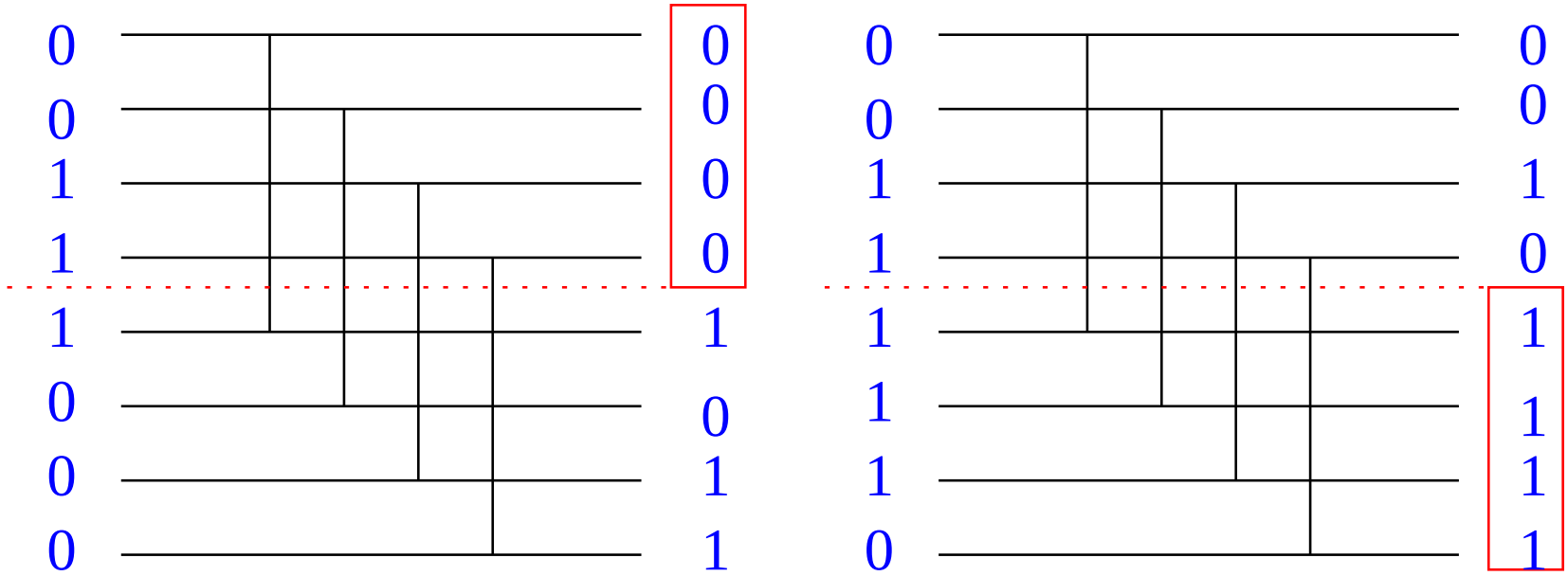
Zasada zero-jedynkowa

...
Z Lematu 1, przy wejściu (zero-jedynkowym) $f(a) =< f(a_1), \ldots, f(a_n) >$, $f(a_i)$ i $f(a_j)$ pojawią się na tych samych wyjściach co odpowiednio a_i i a_j. Ponieważ $f(a_i) = 0$ jest po $f(a_j) = 1$, więc ciąg wyjściowy nie jest posortowany. Sprzeczność! $\square$

Definicja. Ciąg jest *bitoniczny*, jeśli jest połączeniem ciągu niemalejącego z nierosnącym, lub nierosnącego z niemalejącym. Np. $< 1, 4, 6, 8, 3, 2 >$, $< 9, 8, 3, 2, 4, 6 >$.
Zero-jedynkowe ciągi bitoniczne mają postać: $0^i 1^j 0^k$ lub $1^i 0^j 1^k$, dla pewnych $i, j, k \geq 0$.

Half-Cleaner

Definicja. Sieć `Half-Cleaner[`n`]`: Sieć o n wejściach, głębokości 1, dla każdego $i = 1, \ldots, n/2$, i-te wejście połączone komparatorem z wejściem $i + n/2$.



Głębokość: 1, liczba komparatorów: $n/2$.

Half-Cleaner

Lemat. Dla zero-jedynkowego ciągu bitonicznego na wejściu, ciąg wyjściowy sieci `Half-Cleaner[`n`]` ma następującą postać: Obie połowy są ciągami bitonicznymi, oraz górna połowa składa się z samych zer lub dolna – z samych jedynek.

D-d. Rozważmy ciąg wejściowy postaci $0^i 1^j 0^k$.

Podprzypadek $i + k \geq j$: Na wejściach każdego komparatora jest ≥ 1 zero. (Komparator jest niekrótszy niż odstęp między ciągami zer: $j \leq n/2$.) Górna połowa wyjścia zapełni się zerami.

Jeśli wszystkie jedynki były w jednej połowie ciągu wejściowego, to ta połowa była ciągiem bitonicznym i zostanie skopiowana do dolnej połowy na wyjściu.

. . .

Half-Cleaner

Jeśli jedynki są w obu połówkach wejścia, to jedynki z górnej połówki wejścia wypełnią końcówkę dolnej połówki wyjścia a pozostałe pozostaną prefiksem dolnej połówki. Dolna połowka będzie miała postać: ciąg jedynek przedzielony ciągiem zer (ciąg bitoniczny).

Podprzypadek $i + k \leq j$: Na wejściach każdego komparatora jest ≥ 1 jedynka. (Komparator jest niedłuższy niż odstęp między ciągami zer i niekrótszy od każdego z nich.) Dolna połówka zostanie wypełniona jedynkami.

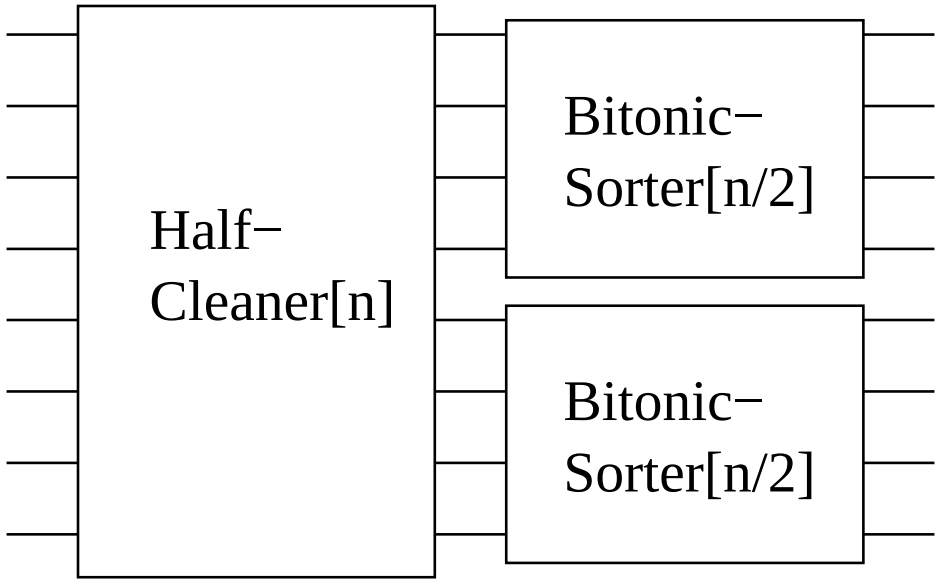
Zera z dolnej połówki wejścia wypełnią końcówkę górnej połówki wyjścia. Górna połówka wyjścia ma postać: ciąg zer przedzielony ciągiem jedynek (ciąg bitoniczny).

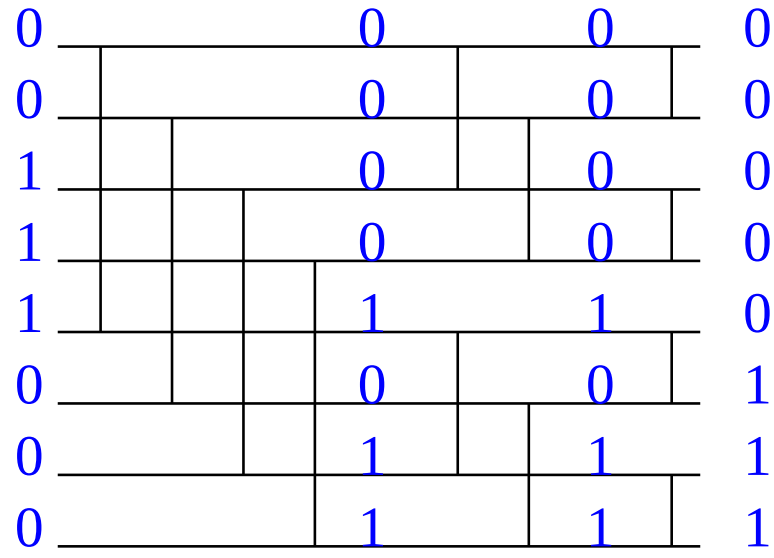
Dla ciągu wejściowego postaci $1^i 0^j 1^k$ – dowód analogiczny.

□

Bitonic-Sorter

Definicja. `Bitonic-Sorter[`n`]`: `Half-Cleaner[`n`]`, na wyjściach którego – dwie równoległe kopie kopie `Bitonic-Sorter[`$n/2$`]` (na górnej i dolnej połówce wyjścia).





Bitonic-Sorter

`Bitonic-Sorter[`n`]` sortuje ciągi bitoniczne:

- `Half-Cleaner[`n`]` dzieli ciąg na dwie połówki, z których każda jest bitoniczna i zapewnia, że wszystkie elementy w dolnej połówce są niemniejsze od wszystkich w górnej.
- każda z połówek jest posortowana przez kopię `Bitonic-Sorter[`$n/2$`]`

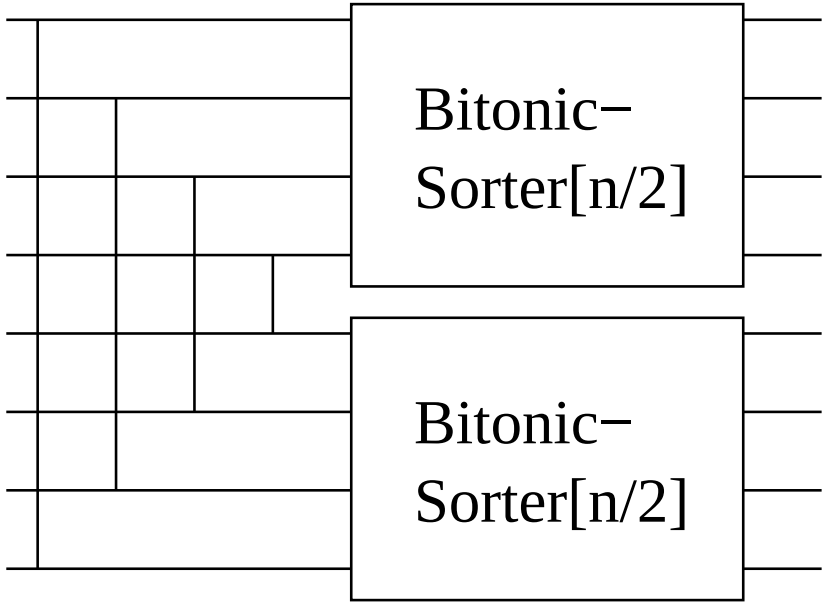
Głębokość: $D(n) = \lg n$ bo:

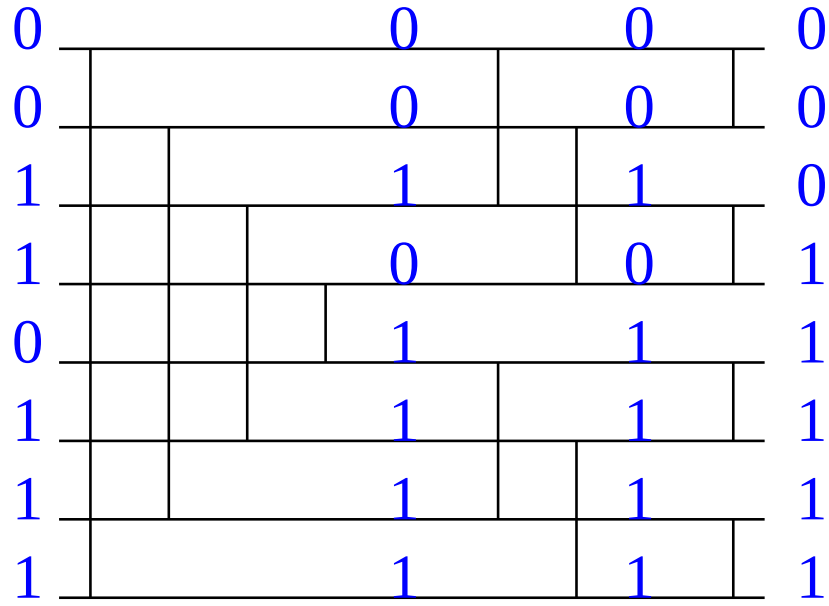
$$D(n) = \begin{cases} 0 & \text{dla } n = 1 \\ D(n/2) + 1 & \text{dla } n = 2^k \text{ oraz } k \geq 1 \end{cases}$$

Liczba komparatorów: $\frac{n}{2} \lg n$.

Merger

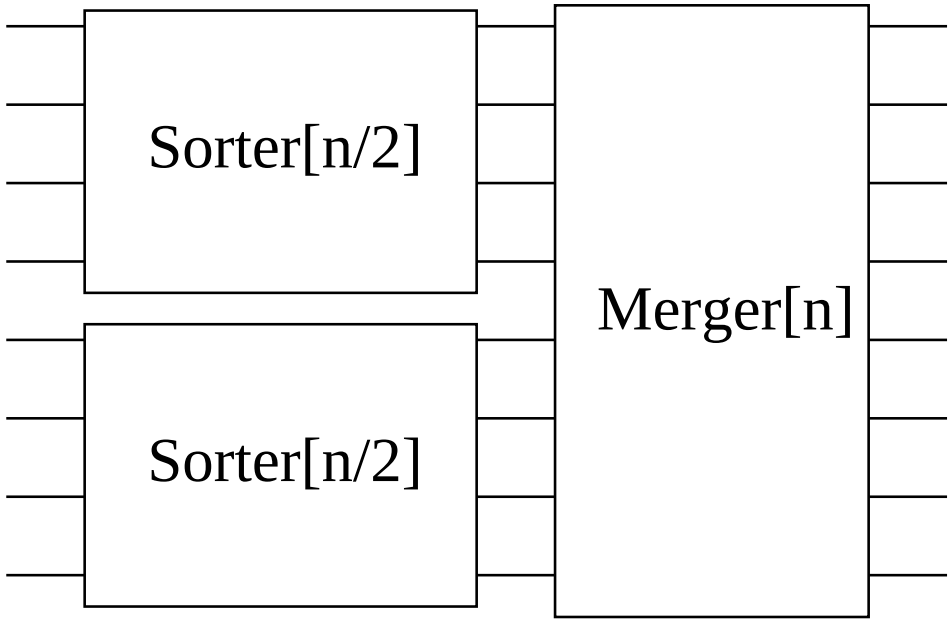
`Merger[`n`]` – scala dwa posortowane ciągi długości $n/2$. Jeśli drugi ciąg odwrócimy i dokleimy do pierwszego, to otrzymamy ciąg bitoniczny.
`Merger[`n`]` – uzyskujemy z `Bitonic-Sorter[`n`]` odwracając kolejność wejść i wyjść w dolnej połówce `Half-Cleaner[`n`]`.



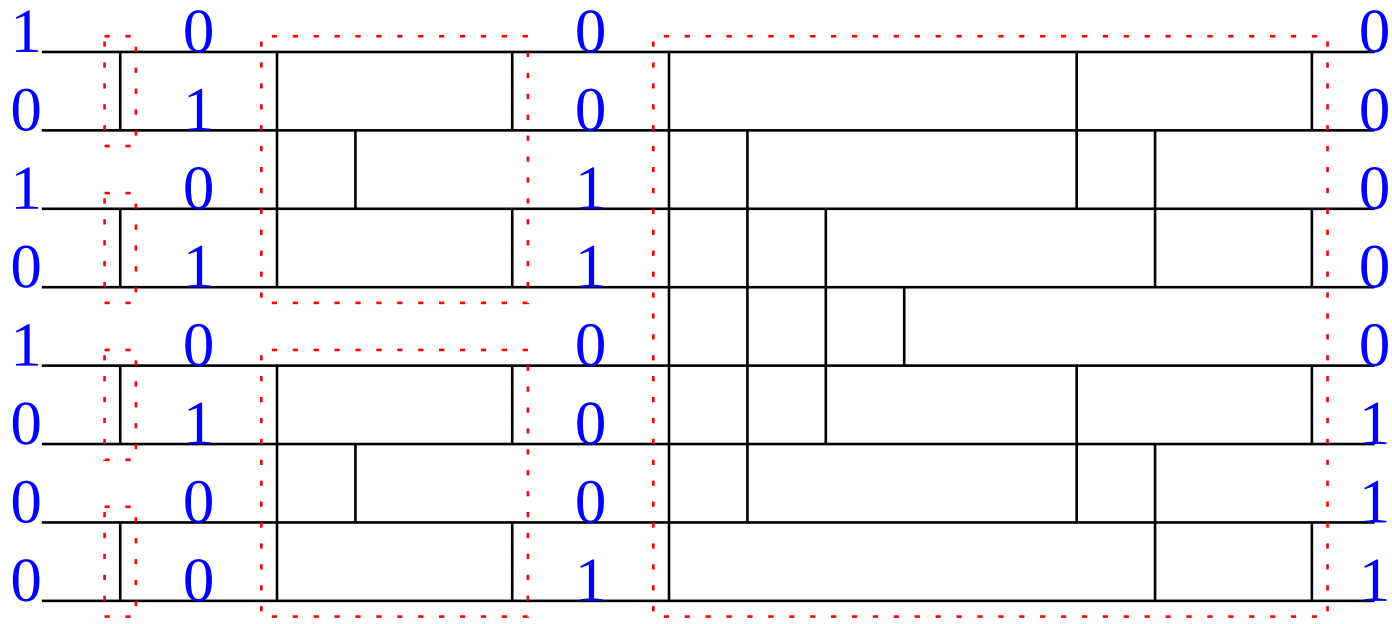


Sorter

Definicja. `Sorter[n]` (bitoniczna sieć sortująca Batchera): Dwie kopie `Sorter[n/2]` sortujące górną i dolną połówkę oraz sieć `Merger[n]` scalająca obie posortowane połówki.



Sorter



Głębokość: $D(n) = \Theta(\lg^2 n)$, bo:

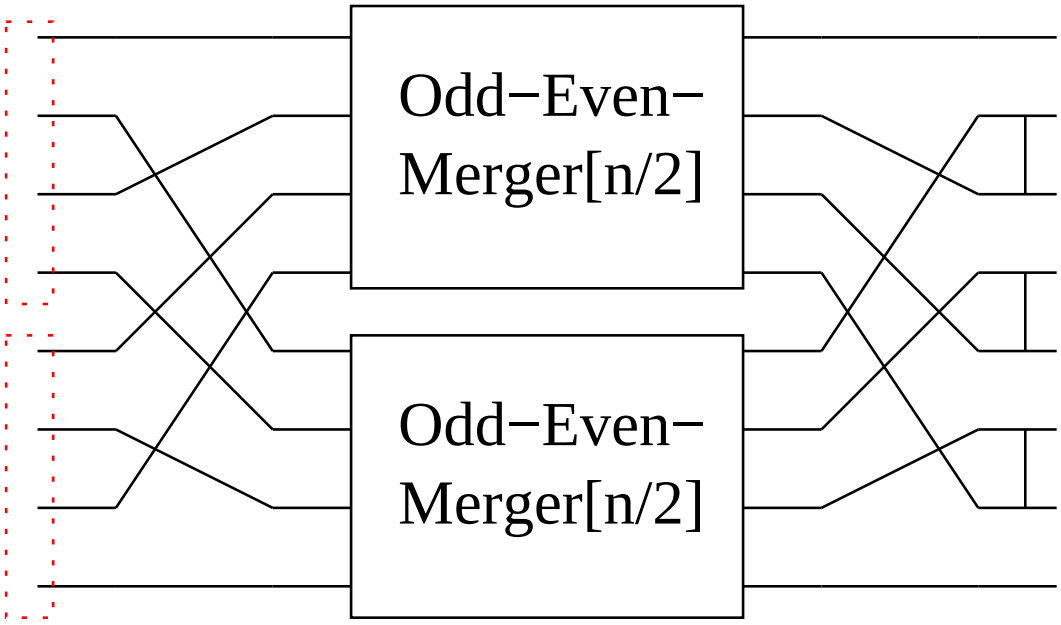
$$D(n) = \begin{cases} 0 & \text{dla } n = 1 \\ D(n/2) + \lg n & \text{dla } n = 2^k \end{cases}$$

Liczba komparatorów: $\Theta(n \lg^2 n)$.

Odd-Even-Merger

Załóżmy, że $n = 2^k$, dla $k > 1$. `Odd-Even-Merger[`n`]` scala posortowane podciągi $< a_1, \ldots, a_{n/2} >$ i $< a_{n/2+1}, \ldots, a_n >$ następująco: dwie kopie `Odd-Even-Merger[`$n/2$`]` scalają pary podciągów $< a_1, a_3, \ldots, a_{n/2-1} >$ z $< a_{n/2+1}, a_{n/2+3}, \ldots, a_{n-1} >$ oraz $< a_2, a_4, \ldots, a_{n/2} >$ z $< a_{n/2+2}, a_{n/2+4}, \ldots, a_n >$, (tak, że elementy obu par pozostają na swoich liniach), a następnie, dla $i = 1, \ldots, n/2 - 1$ umieszczamy komparator między liniami $2i$ i $2i + 1$.
Dla $n = 2^1$ sieć składa się z jednego komparatora.

Odd-Even-Merger



Głębokość: $\lg n$. Liczba komparatorów: $\Theta(n \lg n)$.
(Można jej użyć, podobnie jak bitonicznej sieci scalającej, do konstrukcji sieci sortującej o głębokości $\Theta(\lg^2 n)$ – sieć Batchera odd-even merge sort)

Odd-Even-Merger

Poprawność: (Z zasady zero-jedynkowej) Niech $< a_1, \ldots, a_{n/2} >$ jest postaci $0^k 1^{n/2-k}$, a $< a_{n/2+1}, \ldots, a_n >$ – postaci $0^l 1^{n/2-l}$. Wtedy do "nieparzystej" sieci `Odd-Even-Merge[n/2]` trafi $z_1 = \lceil k/2 \rceil + \lceil l/2 \rceil$ zer a do "parzystej" sieci `Odd-Even-Merge[n/2]` trafi $z_2 = \lfloor k/2 \rfloor + \lfloor l/2 \rfloor$ zer. Ich wyjścia stanowią dwa "przeplecione" ciągi $0^{z_1} 1^{n/2-z_1}$ i $0^{z_2} 1^{n/2-z_2}$. $(z_1 - z_2) \in \{0, 1, 2\}$. Jeśli $z_1 - z_2 = 0$ lub $z_1 - z_2 = 1$, to wyjście jest już posortowane (z ostatnim zerem na odpowiednio parzystej lub nieparzystej pozycji). Jeśli $z_1 - z_2 = 2$, to istnieje jedna para pozycji $2i$, $2i + 1$, taka, że na pozycji $2i$ jest jedynka a na pozycji $2i + 1$ jest zero. Komparator między tymi pozycjami zmieni ciąg na posortowany.

Dolna granica na scalanie

Niech $M(m,n)$ oznacza najmniejszą liczbę komparatorów potrzebnych do scalenia posortowanych ciągów m-elementowego i n-elementowego.

Tw. (Floyd) $M(2n,2n) \geq 2M(n,n)+n$, dla każdego $n \geq 1$.

D-d. Załóżmy, że mamy sieć o wejściu $< z_1, \ldots, z_{4n} >$, która scala posortowane podciągi $< z_1, z_3, \ldots, z_{4n-1} >$ i $< z_2, z_4, \ldots, z_{4n} >$ i ma $M(2n,2n)$ komparatorów. Można przyjąć, że każdy komparator jest skierowany w dół. (Każdą sieć, która ma posortowane wyjście można przerobić na sieć standardową nie zwiększając liczby komparatorów (ćw.).)

...

Dolna granica na scalanie

Każdy komparator $[i : j]$ sieci może być w jednej z trzech klas:

- klasy A, jeśli $i \leq 2n$ i $j \leq 2n$
- klasy B, jeśli $i > 2n$ i $j > 2n$
- klasy C, jeśli $i \leq 2n$ i $j > 2n$

Klasa A musi zawierać $\geq M(n, n)$ komparatorów, bo $< z_{2n+1}, \ldots, z_{4n} >$ może już stanowić posortowany ciąg największych elementów, czyli zadanie sprowadza się do scalenia górnych połówek podciągów przez klasę A. Analogicznie musi być $\geq M(n, n)$ komparatorów w klasie B. Klasa C musi zawierać $\geq n$ komparatorów, bo na przykład dla ciągu wejściowego $< 0, 1, 0, 1, \ldots, 0, 1 >$, n zer trzeba przemieścić z pozycji $\{2n + 1, \ldots, 4n\}$ do pozycji $\{1, \ldots, 2n\}$.

□

Dolna granica na scalanie

Z Tw. wynika: dla $n = 2^k$, $M(n,n) \geq 2M(n/2, n/2) + n/2$, czyli $M(n,n) = \Omega(n \lg n)$ (ćw.).
Uwaga 1: Sekwencyjnie można scalić dwa posortowane ciągi przy użyciu $O(n)$ porównań (np. `Merge` w `Merge-Sort`). Stąd wynika, że sieci komparatorów wymagają *istotnie* więcej porównań.
Uwaga 2: `Merger` i `Odd-Even-Merger` są asymptotycznie optymalne pod wzgl. liczby komparatorów. (również pod wzgl. głębokości, bo nie może być więcej niż $n/2$ komparatorów tej samej głebokości.)

Dolna granica na selekcję

Problem: Chcemy w ciągu n elementów wybrać t najmniejszych na ustalonych t wyjściach. Niech $U_t(n)$ oznacza najmniejszą liczbę kompartatorów w sieci dla tego problemu.

Tw. (Alekseyev) $U_t(n) \geq (n-t)\lceil \log_2(t+1) \rceil$.

D-d. Niech N sieć rozwiązująca problem. Dla każdej krawędzi i sieci N, niech l_i oznacza najmniejszą wartość pojawiającą się na tej krawędzi, jeśli na wejściu mogą się pojawiać wszystkie permutacje liczb $\{1, \ldots, n\}$. Jeśli i jest krawędzią wejściową, to oczywiście $l_i = 1$. Jeśli i i j są krawędziami wchodzącymi do komparatora c, a i' i j' sa odpowiednio górną i dolną krawędzią wychodzącą z c, to:

- $l_{i'} = \min\{l_i, l_j\}$ (oczywiste)
- $l_{j'} \leq l_i + l_j$ (uzasadnienie – następny slajd)

Dolna granica na selekcję

Uzasadnienie $l_{j'} \leq l_i + l_j$: Niech x, y – permutacje, dla których odpowiednio na krawędziach i i j pojawiają się l_i i l_j. Niech x' (odp. y') – ciąg zero-jedynkowy, w którym zera są na pozycjach, na których w x (odp. y) były wartości $\leq l_i$ (odp. $\leq l_j$). Przy wejściu x' (odp. y') na krawędzi i (odp. j) pojawi się 0 (Lemat 1). Niech z' – bitowa koniunkcja x' i y'. z' ma $\xi_{z'} \leq l_i + l_j$ zer (*bo* x' *ma* l_i *zer a* y' *ma* l_j *zer*). Przy wejściu z' na krawędziach i i j pojawią się zera, zatem na wyjściu j' musi być zero. (*Pozycje zer w* z' *są nadzbiorem pozycji zer w* x' *(odp. w* y'*). Stąd tam gdzie pojawią się zera przy wejściu* x' *(odp.* y'*), tam będą też zera przy wejściu* z'*.*) Niech z permutacja, w której wartości $\leq \xi_{z'}$ są tam gdzie zera w z'. Przy wejściu z na wyjściu j' pojawi się (odpowiadająca zeru w z') wartość $\leq \xi_{z'} \leq l_i + l_j$ (Lemat 1).

Dolna granica na selekcję

Przeinterpretujmy działanie sieci następująco: Wszystkie wejścia zawierają 0, a każdy komparator umieszcza swoje mniejsze wejście na górnym wyjściu a większe powiększone o 1 – na dolnym.
Jeśli na krawędzi i pojawia się m_i, to $2^{m_i} \geq l_i$, ponieważ ta nierówność zachodzi dla krawędzi wejściowych i jest zachowana przez każdy komparator, bo:

- $2^{\min\{m_i,m_j\}} \geq \min\{l_i, l_j\}$, oraz
- $2^{\max\{m_i,m_j\}+1} \geq l_i + l_j$

Niech $i_1, \ldots, i_n$ krawędzie wyjściowe. Łaczna liczba komparatorów w sieci wynosi $m_{i_1} + \ldots + m_{i_n}$, bo każdy komparator wnosi $+1$ do tej sumy.

...

Dolna granica na selekcję

Ponieważ sieć wybiera t najmniejszych wartości na ustalonych wyjściach, pozostałe $n-t$ wyjść musi mieć $l_{i_k} \geq t+1$. Stąd $n-t$ wartości m_{i_k} jest $\geq \lceil \log_2(t+1) \rceil$. $\square$

Wnioski: Jeśli np. chcemy rozdzielić elementy na większą i mniejszą połowę, (t.j. selekcja $n/2$ najmniejszych elementów), to potrzebujemy $\Omega(n \lg n)$ komparatorów i (stąd) głębokości $\Omega(\lg n)$. (*Sekwencyjna procedura* `Select` – $O(n)$ *porównań do wyboru mediany i dodatkowe* $O(n)$ *porównań z medianą do podziału na większą i mniejszą połowę.*)
(Uwaga: dla dowolnego $\epsilon > 0$ istnieje sieć (ϵ-*halver*) o stałej głębokości (zależnej tylko od ϵ), która pozostawia $\leq \epsilon n$ spośród $n/2$ najmniejszych elementów w dolnej połowie wyjścia. Wykorzystane w sieci sortującej AKS o głębokości $O(\lg n)$.)