

pomocą metody „zamiatania” efektywnie rozstrzygać, czy dany zbiór odcinków zawiera przecięcia. Dwa pomysły algorytmu znajdowania wypukłej otoczkii zbioru punktów – algorytmy Grahama i Jarvisa – również ilustrują siłę metody zamiatania. Rozdział zakończymy omówieniem efektywnego algorytmu znajdowania pary najmniej odległych punktów w danym zbiorze punktów na płaszczyźnie.

Rozdział 36 dotyczy problemów NP-zupełnych. Do NP-zupełnych zalicza się wiele interesujących problemów obliczeniowych, dla których nie są znane algorytmy rozwiązujące je w czasie wielomianowym. W rozdziale tym przedstawimy metody dowodzenia NP-zupełności. Udowodnimy NP-zupełność kilku klasycznych problemów: sprawdzania czy graf ma cykl Hamiltona, sprawdzania czy formuła logiczna jest spełnialna i sprawdzania czy dany zbiór liczb zawiera podzbiór, którego suma jest równa zadanej wartości. Udowodnimy też, że słynny problem komiwojażera jest NP-zupełny.

W rozdziale 37 pokażemy, jak za pomocą algorytmów aproksymacyjnych znajdować w sposób efektywny przybliżone rozwiązania problemów NP-zupełnych. Dla niektórych problemów NP-zupełnych stosunkowo łatwo jest obliczyć rozwiązania przybliżone bliskie optymalnym, dla innych jednak nawet najlepsze znane algorytmy aproksymacyjne sprawują się coraz gorzej w miarę wzrostu rozmiaru problemu. Są wreszcie i takie problemy, dla których kosztem przedłużenia czasu obliczeń można uzyskiwać coraz lepsze rozwiązania przybliżone. W rozdziale tym opiszemy owe możliwości na przykładzie problemu pokrycia wierzchołkowego, problemu komiwojażera, problemu pokrycia zbioru i problemu sumy podzbioru.

Rozdział 28

Sieci sortujące

W części II rozważaliśmy algorytmy sortowania dla komputerów sekwencyjnych (maszyn ze swobodnym dostępem do pamięci RAM), które mogą w jednej chwili wykonywać tylko jedną operację. W tym rozdziale zajmiemy się algorytmami sortowania dla modelu obliczeń opartego na sieciach porównujących, w których może być wykonywanych jednocześnie wiele porównań.

Sieci porównujące różnią się od modelu RAM w dwóch ważnych aspektach. Po pierwsze, wykorzystując sieci możemy wykonywać tylko porównania. Dlatego algorytm taki jak sortowanie przez zliczanie (patrz podrozdz. 9.2) nie może zostać zaimplementowany z wykorzystaniem sieci porównujących. Po drugie, w przeciwieństwie do modelu RAM, w którym operacje są wykonywane sekwencyjnie, stosując sieci porównujące, wiele operacji można wykonać w tej samej chwili lub „równolegle”. Jak się przekonamy, ta właściwość umożliwia posortowanie n liczb z wykorzystaniem sieci porównujących w czasie mniejszym niż liniowy.

W podrozdziale 28.1 zdefiniujemy sieci porównujące i sortujące. Podamy również naturalną definicję „czasu działania” sieci porównującej za pomocą głębokości sieci. W podrozdziale 28.2 udowodnimy „zasadę zero-jedynkową”, która znacznie ułatwia analizę poprawności sieci sortujących.

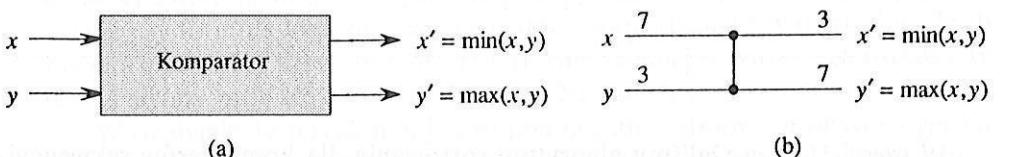
Skonstruujemy efektywną sieć sortującą, będącą w istocie równoległą wersją algorytmu sortowania przez scalanie z podrozdz. 1.3.1. Będziemy ją tworzyć w trzech krokach. W podrozdziale 28.3 opiszymy konstrukcję bitonicznej sieci sortującej, która będzie jej podstawowym składnikiem. W podrozdziale 28.4 zmodyfikujemy nieco bitoniczną sieć sortującą, aby otrzymać sieć scalającą, która scala dwa uporządkowane ciągi w jeden ciąg uporządkowany. Wreszcie w podrozdziale 28.5 zbudujemy z sieci scalających sieć sortującą ciąg n wartości w czasie $O(\lg^2 n)$.

28.1. Sieci porównujące

Sieci sortujące są sieciami porównującymi, które zawsze sortują wartości na wejściu. Rozpoczniemy więc naszą prezentację od sieci porównujących i ich własności. Sieć porównująca składa się tylko z przewodów i komparatorów. Komparator, pokazany na rys. 28.1a, jest urządzeniem z dwoma wejściami x i y oraz dwoma wyjściami x' i y' , „obliczającym” następującą funkcję:

$$x' = \min(x, y)$$

$$y' = \max(x, y)$$



Rys. 28.1. (a) Komparator z wejściami x i y oraz wyjściami x' i y' . (b) Ten sam komparator przedstawiony jako pionowy odcinek. Na wejściu komparatora znajdują się wartości $x = 7$, $y = 3$, a na wyjściu $x' = 3$, $y' = 7$

Graficzna reprezentacja komparatora na rys. 28.1a jest dla naszych celów zbyt obszerna, przyjmijmy więc konwencję przedstawiania komparatorów jako pojedynczych pionowych odcinków, takich jak na rys. 28.1b. Wejścia komparatora znajdują się po lewej stronie, a wyjścia po prawej, przy czym mniejsza wartość pojawia się na górnym przewodzie, a większa na dolnym. Pojedynczy komparator „sortuje” więc ciąg składający się z dwóch liczb pojawiających się na jego wejściu.

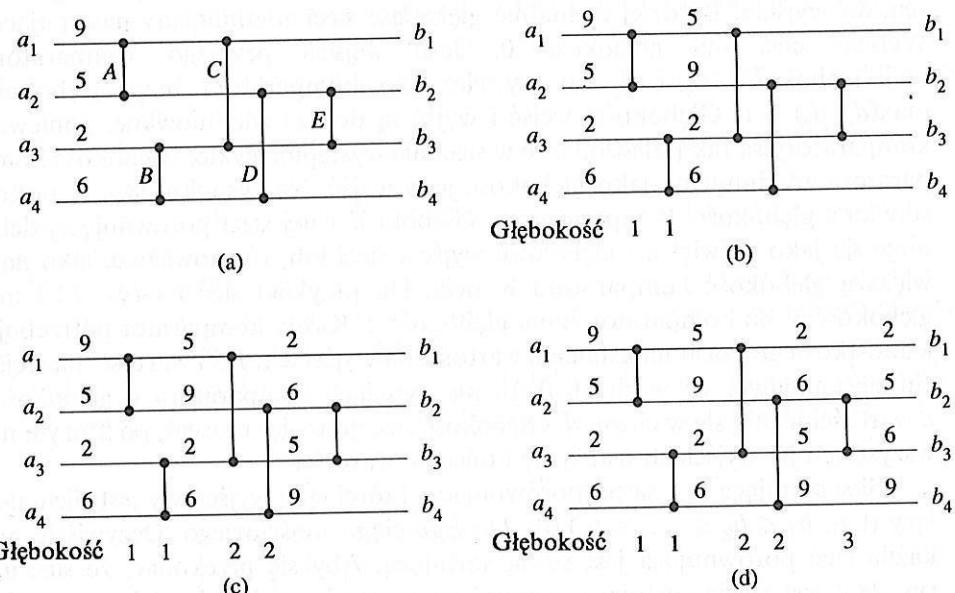
Będziemy zakładać, że każdy komparator działa w czasie $O(1)$. Mówiąc inaczej, przyjmijmy, że czas między podaniem wartości x i y na wejścia komparatora a pojawiением się wartości x' i y' na jego wyjściach jest stały.

W tym rozdziale będziemy przyjmować, że każda sieć porównująca zawiera n wejść a_1, a_2, \dots, a_n , na które są podawane wartości do posortowania, oraz n wyjść b_1, b_2, \dots, b_n , z których odczytujemy wynik. Będziemy także mówić o ciągu wejściowym $\langle a_1, a_2, \dots, a_n \rangle$ oraz ciągu wyjściowym $\langle b_1, b_2, \dots, b_n \rangle$ w odniesieniu do wartości na wejściu i wyjściu.

Na rysunku 28.2 jest przedstawiona sieć porównująca składająca się z komparatorów połączonych za pomocą przewodów. Sieci porównujące o n wejściach będziemy przedstawiać jako n poziomych linii z komparatorami w postaci pionowych odcinków. Należy zaznaczyć, że pojedyncza linia nie reprezentuje jednego przewodu, lecz raczej ciąg przewodów łączących wiele komparatorów. Na przykład najwyższa linia na rys. 28.2 reprezentuje trzy przewody: wejściowy a_1 łączący wejście z komparatorem A , przewód łączący górne wy-

ście komparatora A z wejściem komparatora C oraz przewód wyjściowy b_1 górnego wyjścia komparatora C . Wejścia każdego komparatora są połączone z jednym z n wejściami a_1, a_2, \dots, a_n całej sieci albo z wyjściem innego komparatora. Podobnie wyjścia każdego komparatora są połączone z jednym z n wyjściami b_1, b_2, \dots, b_n całej sieci albo z wejściem innego komparatora. Od połączeń w sieci wymagamy, by ich graf był acykliczny: żadna ścieżka prowadząca z wyjścia pewnego komparatora do wejścia innego nie może przejść dwa razy przez ten sam komparator, wyjście komparatora nie może być połączone bezpośrednio z jego wejściem itp. Dzięki temu możemy zawsze przedstawić sieć porównującą tak jak na rys. 28.2 – z wejściami sieci po lewej stronie i jej wyjściami po prawej stronie; dane „przepływają” w takiej sieci od strony lewej do prawej.

Wartości wyjściowe pojawiają się na wyjściu komparatora dopiero wtedy, gdy otrzyma on na swym wejściu obie wartości. Założymy na przykład, że na wejściu sieci z rys. 28.2a pojawia się w chwili 0 ciąg $\langle 9, 5, 2, 6 \rangle$. Wówczas tylko komparatory A i B mają ustalone wartości na swych wejściach. Przyjmując, że obliczenie wartości wyjściowych komparatora trwa jednostkę czasu, wartości



Rys. 28.2. (a) Sieć porównująca o 4 wejściach i 4 wyjściach, która jest jednocześnie siecią sortującą. W chwili 0 wypisane wartości znajdują się na wejściach. (b) W chwili 1 przetworzone dane wejściowe znajdują się na wyjściach komparatorów A i B, które mają głębokość 1 w sieci. (c) W chwili 2 wartości znajdują się na wyjściach bramek C i D, mających głębokość 2 w sieci. Na wyjściach b_1 i b_4 znajdują się już końcowe wartości, lecz na wyjściach b_2 i b_3 jeszcze nie. (d) W chwili 3 wypisane wartości znajdują się na wyjściu komparatora E. Teraz już i na wyjściach b_2 i b_3 znajdują się ostateczne wartości

na wyjściach komparatorów A i B ustalają się w chwili 1; odpowiednie wyniki są pokazane na rys. 28.2b. Należy zaznaczyć, że wartości na wyjściach komparatorów A i B pojawiają się jednocześnie lub „równolegle”. W chwili 1 komparatory C i D mają ustalone wartości na wejściach (lecz nadal nie ma ich komparator E). W chwili 2 wyznaczają one wartości na swych wyjściach, jak na rys. 28.2c. Komparatory C i D także działają równolegle. Górnego wyjścia komparatora C oraz dolne wyjście komparatora D są podłączone odpowiednio do wyjść b_1 i b_4 . Na tych dwóch wyjściach sieci końcowe wartości pojawiają się więc już w chwili 2. W międzyczasie w chwili 3 pojawiają się odpowiednie wartości na jego wyjściach (patrz rys. 28.2d). Te wartości są więc też na wyjściach b_2 i b_3 , co daje końcowy ciąg wyjściowy sieci $\langle 2, 5, 6, 9 \rangle$.

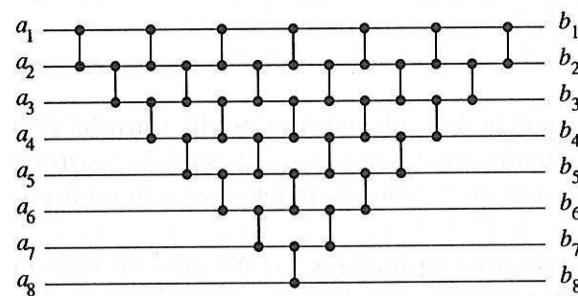
Przy powyższym założeniu, że wartości na wyjściach komparatora ustalają się po upływie jednostki czasu, możemy zdefiniować „czas działania” sieci porównującej, tzn. czas jaki upływa od podania wszystkich wartości na wejściach do chwili, w której na wszystkich wyjściach ustalą się końcowe wartości. Mówiąc nieformalnie, wielkość ta jest równa największej możliwej liczbie komparatorów, przez które może „przejść” informacja wejściowa od pewnego wejścia do wyjścia. Bardziej formalnie głębokość sieci zdefiniujemy następująco. Wejście sieci ma głębokość 0. Jeśli wejścia pewnego komparatora mają głębokości d_x i d_y , to wyjścia tego komparatora mają głębokość $\max(d_x, d_y) + 1$. Głębokości wejść i wyjść są dobrze zdefiniowane, ponieważ komparatory są tak połączone, że w sieci nie występują cykle. Głębokość komparatora zdefiniujemy jako głębokość jego wyjścia. Na rysunku 28.2 są przedstawione głębokości komparatorów. Głębokość całej sieci porównującej definiuje się jako największą głębokość wyjścia sieci lub, równoważnie, jako największą głębokość komparatora w sieci. Na przykład sieć na rys. 28.2 ma głębokość 3, bo komparator E ma głębokość 3. Każdy komparator potrzebuje jednostkowego czasu na ustalenie wartości na wyjściach. Jeśli wartości na wejściu sieci pojawią się w chwili 0, to na wyjściach komparatora o głębokości d wartości ustalą się w chwili d . Głębokość sieci jest więc czasem, po którym na wszystkich jej wyjściach ustalą się końcowe wartości.

Sieć sortująca jest siecią porównującą, której ciąg wyjściowy jest niemalejący (tzn. $b_1 \leq b_2 \leq \dots \leq b_n$) dla *każdego* ciągu wejściowego. Oczywiście nie każda sieć porównująca jest siecią sortującą. Aby się przekonać, że sieć na rys. 28.2 jest siecią sortującą, zauważmy, że w chwili 1 najmniejsza spośród 4 wartości wejściowych znajduje się albo na górnym wyjściu komparatora A , albo na górnym wyjściu komparatora B . W chwili 2 musi się ona znaleźć na górnym wyjściu komparatora C . Podobnie największa z 4 wartości wejściowych znajduje się w chwili 2 na dolnym wyjściu komparatora D . Jedyne co pozostało dla komparatora E to zagwarantowanie, że dwie środkowe wartości zostaną ułożone we właściwej kolejności, co odbywa się w chwili 3.

Sieć porównująca jest podobna do procedury, która ustala kolejność wykonywanych porównań, lecz w przeciwnieństwie do procedury jej fizyczny rozmiar zależy od liczby wejść i wyjść. Dlatego też będziemy raczej opisywać „rodziny” sieci porównujących. Celem tego rozdziału jest na przykład zaprojektowanie rodziny SORTER złożonej z efektywnych sieci sortujących. Pojedyncze egzemplarze sieci z rodziny sieci porównujących określamy za pomocą nazwy rodziny oraz liczby wejść sieci (równiej liczbie wyjść); na przykład sieć o n wejściach i n wyjściach należącą do rodziny SORTER będziemy oznaczać przez SORTER $[n]$.

ZADANIA

- 28.1-1. Podaj wartości, jakie pojawiają się na wszystkich wejściach i wyjściach komparatorów sieci z rys. 28.2, jeśli na wejściu podamy ciąg $\langle 9, 6, 5, 2 \rangle$.
- 28.1-2. Niech n będzie dokładną potęgą 2. Pokaż, jak skonstruować sieć porównującą o n wejściach oraz n wyjściach i głębokości $\lg n$, w której na górnym wyjściu będzie pojawiać się wartość najmniejsza, a na dolnym wyjściu – wartość największa dla dowolnego ciągu wejściowego.
- 28.1-3. Profesor Nielsen twierdzi, że jeśli dodamy gdziekolwiek komparator do sieci sortującej, to sieć nadal będzie poprawnie sortowała ciągi wejściowe. Wykaż, że profesor nie ma racji, dodając komparator do sieci na rys. 28.2 w taki sposób, że nowa sieć nie będzie poprawnie sortować wszystkich permutacji.
- 28.1-4. Udosowodnij, że każda sieć sortująca o n wejściach ma głębokość co najmniej $\lg n$.
- 28.1-5. Udosowodnij, że liczba komparatorów w każdej sieci sortującej jest $\Omega(n \lg n)$.
- 28.1-6. Udosowodnij, że sieć porównująca na rys. 28.3 jest w istocie siecią sortującą. Opisz związek jej struktury z algorytmem sortowania przez wstawianie (patrz podrozdz. 1.1).
- 28.1-7. Sieć sortującą o n wejściach składającą się z c komparatorów można reprezentować jako ciąg c par liczb całkowitych z przedziału od 1 do n .



Rys. 28.3. Sieć sortującą opartą na algorytmie sortowania przez wstawianie

Jeżeli dwie pary zawierają wspólną liczbę, to kolejność odpowiednich komparatorów w sieci ustalamy według kolejności par na liście. Zaprojektuj algorytm (sekwencyjny) działający w czasie $O(n + c)$, obliczający głębokość sieci zadanej w powyższej reprezentacji.

28.1-8. Przypuśćmy, że oprócz zwykłych komparatorów możemy także używać komparatorów „odwróconych”, w których na dolnym wyjściu pojawia się mniejsza, a na górnym większa z liczb wejściowych. Zaproponuj metodę umożliwiającą przekształcenie dowolnej sieci sortującej składającej się ze zwykłych oraz odwróconych komparatorów na sieć o takiej samej liczbie komparatorów, ale bez komparatorów odwróconych. Wykaż, że zaproponowana przez Ciebie metoda jest poprawna.

28.2. Zasada zero-jedynkowa

Zasada zero-jedynkowa mówi, że jeśli sieć sortująca działa poprawnie dla wszystkich ciągów składających się tylko z zer i jedynek, to poprawnie sortuje wszystkie ciągi wartości. (Wartości mogą być liczbami całkowitymi, rzeczywistymi lub należeć do dowolnego zbioru liniowo uporządkowanego). Dzięki zasadzie zero-jedynkowej, konstruując sieci sortujące i inne sieci porównujące, będziemy mogli się skupić na funkcjonowaniu tych sieci dla ciągów składających się tylko z wartości 0 albo 1. Jeśli skonstruujemy sieć sortującą i udowodnimy, że poprawnie sortuje wszystkie ciągi zero-jedynkowe, to wystarczy odwołać się do zasady zero-jedynkowej, aby ustalić poprawność tej sieci dla dowolnych ciągów.

Dowód zasady zero-jedynkowej opiera się na pojęciu funkcji niemalejącej (monotonicznie rosnącej; patrz podrozdz. 2.2).

LEMAT 28.1.

Jeśli sieć porównująca dla ciągu wejściowego $a = \langle a_1, a_2, \dots, a_n \rangle$ wyznacza ciąg wyjściowy $b = \langle b_1, b_2, \dots, b_n \rangle$, to dla dowolnej funkcji niemalejącej f ta sama sieć z ciągiem $f(a) = \langle f(a_1), f(a_2), \dots, f(a_n) \rangle$ na wyjściu wyznacza ciąg wyjściowy $f(b) = \langle f(b_1), f(b_2), \dots, f(b_n) \rangle$

DOWÓD

Udowodnimy najpierw, że jeśli f jest funkcją niemalejącą, to dla wartości $f(x)$ i $f(y)$ na wyjściu pojedynczy komparator wyznacza na wyjściu wartość $f(\min(x, y))$ i $f(\max(x, y))$. Następnie posłużymy się indukcją w celu udowodnienia lematu.

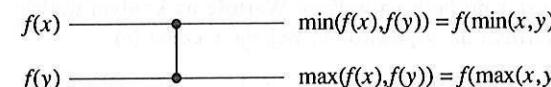
Załóżmy, że na wyjściu komparatora są liczby x i y . Na górnym wyjściu komparatora pojawia się wartość $\min(x, y)$, a na dolnym – wartość $\max(x, y)$. Niech teraz na wyjściu komparatora będą liczby $f(x)$ i $f(y)$, tak jak na

28.2. ZASADA ZERO-JEDYNKOWA

rys. 28.4. Na górnym wyjściu pojawia się wówczas wartość $\min(f(x), f(y))$, a na dolnym – wartość $\max(f(x), f(y))$. Funkcja f jest niemalejąca, więc $x \leq y$ pociąga za sobą $f(x) \leq f(y)$. W rezultacie prawdziwe są następujące tożsamości:

$$\min(f(x), f(y)) = f(\min(x, y))$$

$$\max(f(x), f(y)) = f(\max(x, y))$$



Rys. 28.4. Wynik działania komparatora z dowodu lematu 28.1. Funkcja f jest niemalejąca

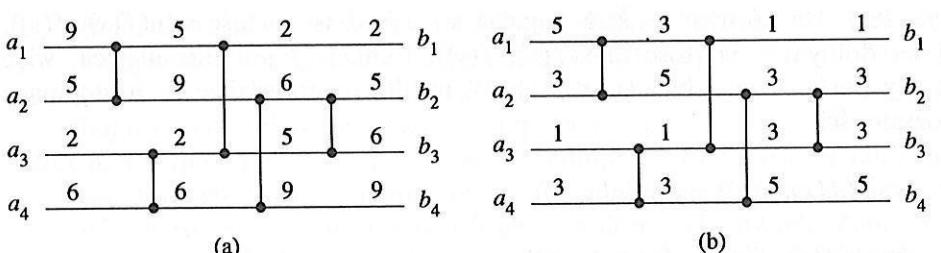
Stąd wynika, że jeśli na wejścia komparatora będą podane wartości $f(x)$ i $f(y)$, to na wyjściach pojawią się odpowiednio: $f(\min(x, y))$ i $f(\max(x, y))$.

Użyjemy teraz indukcji względem głębokości każdego połączenia sieci porównującej w celu wykazania faktu mocniejszego niż teza lematu: przyjmijmy, że na pewnym połączeniu pojawia się wartość a_i , gdy na wejście sieci jest podawany ciąg a , wówczas dla ciągu wejściowego $f(a)$ na tym połączeniu pojawia się wartość $f(a_i)$. Stwierdzenie to dotyczy także wyjść, jego dowód będzie więc jednocześnie dowodem tezy lematu.

Początek indukcji jest trywialny: wejście sieci jest na głębokości 0. Jeśli na wejścia sieci jest podany ciąg $f(a)$, to oznacza, że na i -tym wejściu jest wartość $f(a_i)$. Aby wykonać krok indukcyjny, rozważmy komparator o głębokości $d \geq 1$. Oznacza to, że wyjście komparatora ma głębokość d , wejścia tego komparatora mają więc głębokość istotnie mniejszą niż d . Z założenia indukcyjnego wynika zatem, że jeśli na wyjściach komparatora pojawiają się wartości a_i i a_j dla ciągu wejściowego a , to dla ciągu $f(a)$ na wyjściu sieci na wejściach komparatora pojawiają się wartości $f(a_i)$ oraz $f(a_j)$. Korzystając z wcześniejszego wykazanego faktu, wnioskujemy, że na wyjściach tego komparatora pojawiają się wartości $f(\min(a_i, a_j))$ i $f(\max(a_i, a_j))$. Jeśli na wejście sieci jest podany ciąg a , to na wyjściach tego komparatora pojawiają się wartości $\min(a_i, a_j)$ i $\max(a_i, a_j)$; lemat został więc udowodniony. ◆

Przykład wykorzystania lematu 28.1 jest pokazany na rys. 28.5, na którym widać wynik zastosowania niemalejącej funkcji $f(x) = \lceil x/2 \rceil$ do ciągu wejściowego i sieci porównującej z rys. 28.2. Wartość na każdym wejściu jest wynikiem zastosowania funkcji f do wartości z odpowiedniego wejścia na rys. 28.2.

Jeśli sieć porównująca jest siecią sortującą, to z lematu 28.1 wynika następujące ważne twierdzenie.



Rys. 28.5. (a) Sieć sortująca z rys. 28.2 z ciągiem $\langle 9, 5, 2, 6 \rangle$ na wejściu. (b) Ta sama sieć po zastosowaniu funkcji niemalejącej $f(x) = \lceil x/2 \rceil$ do liczb na wejściu. Wartość na każdym wejściu jest wynikiem zastosowania funkcji f do wartości na odpowiednim wejściu w części (a)

TWIERDZENIE 28.1. (Zasada zero-jedynkowa)

Jeśli sieć porównująca o n wejściach poprawnie sortuje wszystkie 2^n ciągi zer i jedynek, to sortuje poprawnie dowolne ciągi liczb.

DOWÓD

Załóżmy przeciwnie, że sieć sortuje poprawnie wszystkie ciągi zero-jedynkowe, ale istnieje pewien ciąg liczb, dla którego nie działa właściwie. Mówiąc inaczej, istnieje ciąg wejściowy $\langle a_1, a_2, \dots, a_n \rangle$ zawierający elementy a_i oraz a_j takie, że $a_i < a_j$, ale w ciągu wyjściowym sieci a_j znajduje się przed a_i . Zdefiniujmy niemalejącą funkcję f następująco:

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

Jeśli ciągiem wejściowym sieci jest $\langle a_1, a_2, \dots, a_n \rangle$, to a_j znajduje się w ciągu wyjściowym przed a_i ; z lematu 28.1 wynika więc, że jeśli na wejściu sieci znajduje się ciąg $\langle f(a_1), f(a_2), \dots, f(a_n) \rangle$, to wtedy w ciągu wyjściowym $f(a_j)$ będzie przed $f(a_i)$. Stanowi to jednak sprzeczność z faktem, że sieć poprawnie sortuje ciąg zero-jedynkowy $\langle f(a_1), f(a_2), \dots, f(a_n) \rangle$, ponieważ $f(a_j) = 1$, a $f(a_i) = 0$. ◆

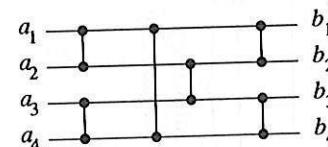
ZADANIA

28.2-1. Udowodnij, że w wyniku zastosowania funkcji niemalejącej do ciągu posortowanego otrzymamy ciąg posortowany.

28.2-2. Udowodnij, że sieć porównująca poprawnie sortuje n -elementowy ciąg $\langle n, n-1, \dots, 1 \rangle$ wtedy i tylko wtedy, gdy poprawnie sortuje wszystkie $n-1$ ciągów zero-jedynkowych $\langle 1, 0, 0, \dots, 0, 0 \rangle, \langle 1, 1, 0, \dots, 0, 0 \rangle, \dots, \langle 1, 1, 1, \dots, 1, 0 \rangle$.

28.2-3. Użyj zasady zero-jedynkowej do wykazania, że sieć porównująca przedstawiona na rys. 28.6 jest siecią sortującą.

28.2-4. Sformułuj i udowodnij odpowiednik zasady zero-jedynkowej dla modelu drzew decyzyjnych. (Wskazówka: Zadbaj o właściwe potraktowanie równości).



Rys. 28.6. Sieć sortująca dla ciągów 4-elementowych

28.2-5. Udowodnij, że sieć sortująca o n wejściach musi zawierać co najmniej jeden komparator między i -tą a $(i+1)$ -szą linią dla $i = 1, 2, \dots, n-1$.

28.3. Bitoniczna sieć sortująca

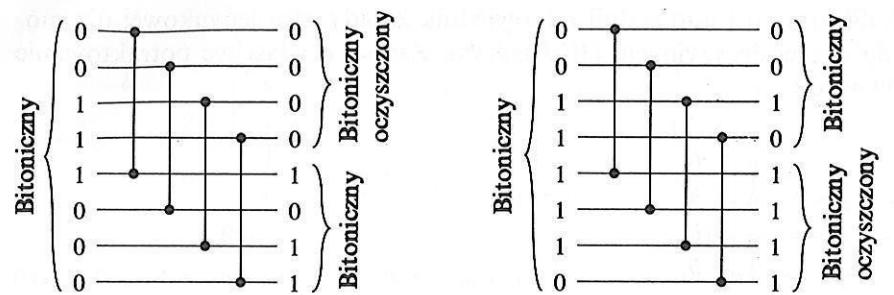
Pierwszym krokiem podczas konstruowania sieci sortującej będzie zaprojektowanie sieci porównującej, która sortuje wszystkie ciągi bitoniczne, czyli ciągi, które albo najpierw są nierosnące, a potem niemalejące, albo najpierw niemalejące, a potem nierosnące. Ciagi $\langle 1, 4, 6, 8, 3, 2 \rangle$ oraz $\langle 9, 8, 3, 2, 4, 6 \rangle$ są bitoniczne. Zero-jedynkowe ciągi bitoniczne mają bardzo prostą postać: albo $0^i 1^j 0^k$ albo $1^i 0^j 1^k$, dla pewnych $i, j, k \geq 0$. Oczywiście ciąg niemalejący lub nierosnący jest także bitoniczny.

Bitoniczna sieć sortująca, którą zbudujemy, będzie siecią porównującą, sortującą bitoniczne ciągi zer i jedynek. Zadanie 28.3-6 polega na wykazaniu, że ta sieć poprawnie sortuje bitoniczne ciągi o dowolnych wartościach.

Sieć półczyszcząca

Na sieć sortującą ciągi bitoniczne składa się wiele połączonych szeregowo sieci półczyszczących. Każda sieć półczyszcząca ma głębokość 1; i -te wejście jest połączone za pomocą komparatora z wejściem $i+n/2$ dla $i = 1, 2, \dots, n/2$ (zakładamy, że n jest parzyste). Na rysunku 28.7 widać sieć półczyszczącą HALF-CLEANER[8] o 8 wejściach i 8 wyjściach.

Jeśli bitoniczny ciąg zer i jedynek jest podawany na wejście sieci półczyszczącej, to na jej wyjściu w górnej połowie znajdują się mniejsze wartości, w dolnej większe, a obie połówki są bitoniczne. W istocie jedna z nich jest wtedy oczyszczona – czyli składa się z samych tylkó zer lub samych jedynek – stąd nazwa sieci półczyszczącej. (Pamiętajmy, że ciągi oczyszczone są bitoniczne). Powyższe własności sieci półczyszczących zostały sformułowane w następującym lemacie.



Rys. 28.7. Sieć porównująca HALF-CLEANER[8]. Są tu pokazane dwa różne ciągi wejściowe i wyjściowe. Zakładamy, że na wejściu sieci jest ciąg bitoniczny. Wtedy na wyjściu sieci półczyszczącej każdy element w górnej połowie jest nie większy niż każdy element w dolnej połowie. Co więcej, obie połowy są bitoniczne i co najmniej jedna z nich jest oczyszczona.

LEMAT 28.3.

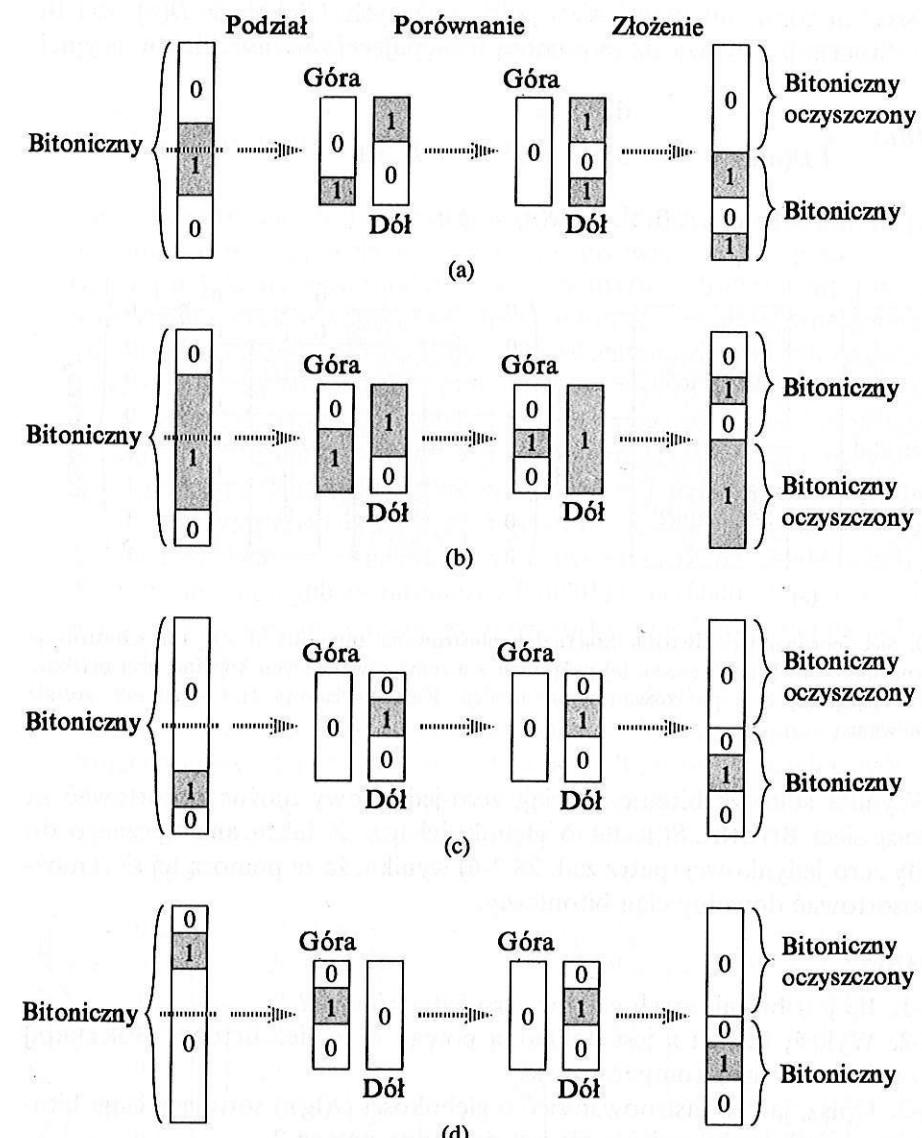
Jeśli na wejście sieci półczyszczącej jest podany bitoniczny ciąg zer i jedynek, to ciąg wyjściowy ma następujące własności: obie połowy – góra i dolna – są bitoniczne, każdy element w górnej połowie jest nie większy niż dowolny element w dolnej połowie oraz co najmniej jedna połowa jest oczyszczona.

DOWÓD

W sieci porównującej HALF-CLEANER[n] są porównywane wartości na wejściach o numerach i oraz $i + n/2$ dla $i = 1, 2, \dots, n/2$. Bez straty ogólności możemy założyć, że ciąg wejściowy jest postaci $00 \dots 011 \dots 100 \dots 0$. (Sytuacja, w której ciąg wejściowy jest postaci $11 \dots 100 \dots 011 \dots 1$ jest symetryczna). Możliwe są trzy przypadki, w zależności od tego, w który ciąg zer lub jedynek „trafia” punkt $n/2$, a jeden z nich (ten, w którym punkt środkowy przypada w bloku jedynek) rozbiija się na dwa kolejne osobne przypadki. Wszystkie cztery sytuacje są pokazane na rys. 28.8. W każdej z nich są spełnione tezy lematu. ♦

Bitoniczna sieć sortująca

Składając sieci półczyszczące rekurencyjnie, tak jak na rys. 28.9, otrzymujemy sieć sortującą ciągi bitoniczne. Najpierw w skład sieci BITONIC-SORTER [n] wchodzi sieć HALF-CLEANER [n], na której wyjściu, zgodnie z lematem 28.2, pojawiają się dwa ciągi bitoniczne o dwukrotnie mniejszej długości, takie że każdy element w górnej połowie sieci jest nie większy niż dowolny element w jej dolnej połowie (patrz rys. 28.9). Wystarczy więc użyć dwóch kopii sieci BITONIC-SORTER [n/2], aby rekurencyjnie posortować obie połowy ciągu wejściowego. Na rysunku 28.9a rekurencja została ukazana w sposób jawnym, a na

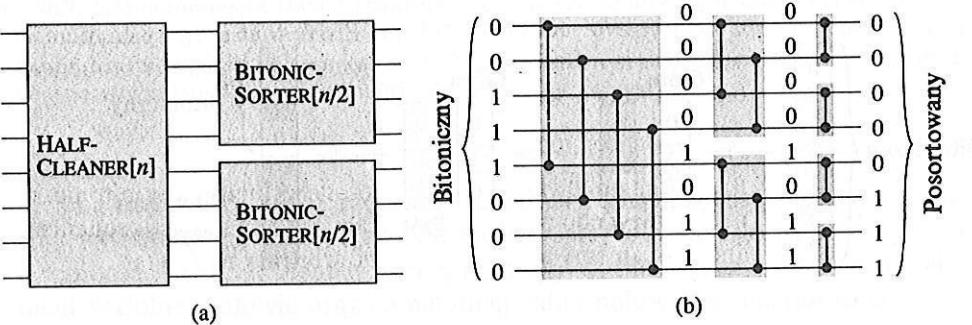


Rys. 28.8. Możliwe porównania w sieci HALF-CLEANER[n]. O ciągu wejściowym zakładamy, że jest bitonicznym ciągiem zer i jedynek. Bez straty ogólności możemy założyć, że jest on postaci $00 \dots 011 \dots 100 \dots 0$. Podciągi zer są białe, a podciągi składające się z jedynek szare. Wejścia są podzielone na połowy, tak że dla $i = 1, 2, \dots, n/2$ wejścia o numerze i oraz $i + n/2$ zostają porównane. (a)-(b) Przypadki, w których połowa przypada w ciągu jedynek. (c)-(d) Przypadki, w których połowa przypada w ciągu zer. We wszystkich przypadkach każdy element w górnej połowie jest nie większy niż każdy element w dolnej połowie, obie połowy są bitoniczne oraz co najmniej jedna połowa jest oczyszczona.

rys. 28.9b – została „rozwinięta” i widać, że bitoniczna sieć sortująca składa się z szeregu coraz mniejszych sieci półczyszczących. Głębokość $D(n)$ sieci BITONIC-SORTER[n] wyraża się za pomocą następującej zależności rekurencyjnej:

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

której rozwiązaniem jest funkcja $D(n) = \lg n$.



Rys. 28.9. Sieć porównująca BITONIC-SORTER[n] zilustrowana tutaj dla $n = 8$. (a) Konstrukcja rekurencyjna: sieć HALF-CLEANER [n] połączona z dwoma równoległymi kopiami sieci BITONIC-SORTER [n/2]. (b) Sieć po rozwinięciu rekurencji. Każda składowa HALF-CLEANER została zaciemniona

Wynika stąd, że bitoniczny ciąg zero-jedynkowy można posortować za pomocą sieci BITONIC-SORTER o głębokości $\lg n$. Z faktu analogicznego do zasady zero-jedynkowej (patrz zad. 28.3-6) wynika, że za pomocą tej sieci można posortować dowolny ciąg bitoniczny.

ZADANIA

- 28.3-1. Ile jest bitonicznych ciągów zero-jedynkowych?
- 28.3-2. Wykaż, że jeśli n jest dokładną potągą 2, to sieć BITONIC-SORTER[n] zawiera $\Theta(n \lg n)$ komparatorów.
- 28.3-3. Opisz, jak skonstruować sieć o głębokości $O(\lg n)$ sortującą ciągi bitoniczne, jeśli liczba wejść n nie jest dokładną potągą 2.
- 28.3-4. Wykaż, że jeśli na wejściu sieci półczyszczącej jest dowolny ciąg bitoniczny, to ciąg wyjściowy ma następujące własności: obie połowy – górna i dolna – są bitoniczne oraz każdy element w górnej połowie jest nie większy niż dowolny element w dolnej połowie.
- 28.3-5. Niech dane będą dwa ciągi zer i jedynek. Udowodnij, że jeśli każdy element jednego z ciągów jest nie mniejszy niż dowolny element drugiego, to jeden z ciągów jest oczyszczony.

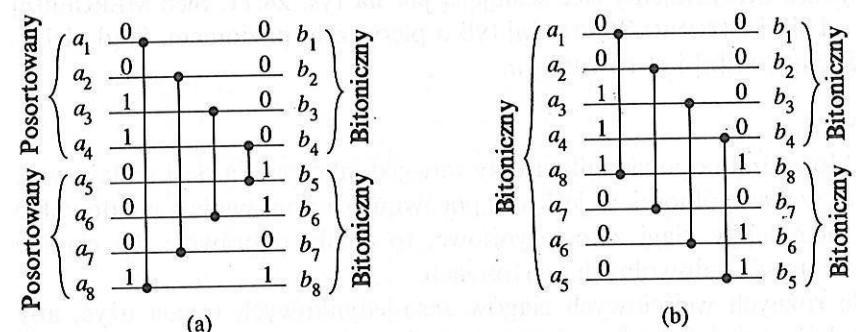
28.3-6. Udowodnij następujący odpowiednik zasady zero-jedynkowej dla bitonicznych sieci sortujących: jeśli sieć sortuje dowolny bitoniczny ciąg zero-jedynkowy, to poprawnie sortuje dowolny ciąg bitoniczny.

28.4. Sieć scalająca

Nasza sieć sortująca będzie się składać z sieci scalających, które służą do scalania dwóch posortowanych ciągów w jeden posortowany ciąg wyjściowy. Sieć MERGER [n] uzyskamy jako modyfikację sieci BITONIC-SORTER [n]. Podobnie jak w przypadku bitonicznej sieci sortującej, udowodnimy poprawność sieci scalającej tylko dla ciągów zero-jedynkowych. W zadaniu 28.4-1 należy pokazać, jak można rozszerzyć ten dowód na przypadek ciągów dowolnych wartości.

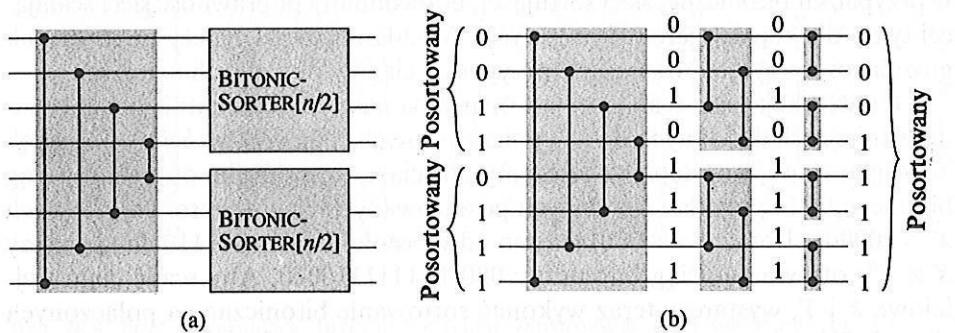
Konstrukcja sieci scalającej jest oparta na następującym intuicyjnym rozumowaniu. Jeśli dla danych dwóch posortowanych ciągów odwrócić porządek w drugim ciągu, a następnie połączymy oba ciągi, to w wyniku otrzymamy ciąg bitoniczny. Na przykład dla danych posortowanych ciągów zero-jedynkowych $X = 00000111$ oraz $Y = 00001111$, po odwróceniu Y ($Y^R = 11110000$) łączymy X z Y^R , otrzymując ciąg bitoniczny 0000011111110000 . Aby scalić ciągi wejściowe X i Y , wystarczy teraz wykonać sortowanie bitoniczne na połączonych ciągach X i Y^R .

Sieć MERGER[n] można skonstruować, modyfikując pierwszą sieć półczyszczącą w sieci BITONIC-SORTER[n]. Pomysł polega na niejawnym odwróceniu drugiej połowy ciągu wejściowego. Dla danych posortowanych ciągów $\langle a_1, a_2, \dots, a_{n/2} \rangle$ oraz $\langle a_{n/2+1}, a_{n/2+2}, \dots, a_n \rangle$, które chcemy scalić, wystarczy nam wynik bitonicznego posortowania ciągu $\langle a_1, a_2, \dots, a_{n/2}, a_n, a_{n-1}, \dots, a_{n/2+1} \rangle$.



Rys. 28.10. Porównanie pierwszego poziomu sieci MERGER[n] oraz sieci HALF-CLEANER[n]. (a) Na pierwszym poziomie w sieci MERGER[n] z dwóch monotonicznych ciągów $\langle a_1, a_2, \dots, a_{n/2} \rangle$ oraz $\langle a_{n/2+1}, a_{n/2+2}, \dots, a_n \rangle$ powstają dwa ciągi bitoniczne $\langle b_1, b_2, \dots, b_{n/2} \rangle$ oraz $\langle b_{n/2+1}, b_{n/2+2}, \dots, b_n \rangle$. (b) Równoważna operacja w sieci HALF-CLEANER[n]. Z ciągu bitonicznego $\langle a_1, a_2, \dots, a_{n/2}, a_n, a_{n-1}, \dots, a_{n/2+1} \rangle$ powstają dwa ciągi bitoniczne $\langle b_1, b_2, \dots, b_{n/2} \rangle$ oraz $\langle b_n, b_{n-1}, \dots, b_{n/2+1} \rangle$

Pierwsza sieć półczyszcząca w sieci BITONIC-SORTER $[n]$ porównuje wejścia i z $n/2 + i$ dla $i = 1, 2, \dots, n/2$, wobec tego w sieci scalającej odpowiedni fragment sieci porównuje wejścia o numerach i oraz $n - i + 1$ (patrz rys. 28.10). W tym nowym rozwiążaniu kryje się pewna subtelność: dolna połowa ciągu wychodzącego z pierwszej fazy sieci scalającej jest odwrócona w stosunku do wyjścia ze zwykłej sieci półczyszczącej. Wystarczy jednak zauważyc, że w wyniku odwrócenia ciągu bitonicznego otrzymamy znowu ciąg bitoniczny, obie połowy można więc następnie posortować bitonicznie, bo spełniają one własności z lematu 28.3.



Rys. 28.11. Sieć scalająca dwa posortowane ciągi wejściowe w jeden posortowany ciąg wyjściowy. Sieć MERGER $[n]$ różni się od sieci BITONIC-SORTER $[n]$ tylko tym, że zamiast pierwszego fragmentu HALF-CLEANER $[n]$ porównuje wejścia i z $n - i + 1$ dla $i = 1, 2, \dots, n/2$. Tutaj $n = 8$. (a) Sieć rozłożona na pierwszy poziom porównań, po której następują dwie równoległe kopie sieci BITONIC-SORTER $[n/2]$. (b) Ta sama sieć z rozwinietą rekurencją. Kolejne poziomy zostały zaciemnione. Przy połączeniach znajdują się przykładowe wartości liczbowe

W wyniku otrzymujemy sieć scalającą jak na rys. 28.11. Sieć MERGER $[n]$ różni się od sieci BITONIC-SORTER $[n]$ tylko pierwszym poziomem. Stąd głębokość sieci MERGER $[n]$ jest równa $\lg n$.

ZADANIA

- 28.4-1. Udowodnij odpowiednik zasady zero-jedynkowej dla sieci scalających. Wykaż w szczególności, że jeśli sieć porównująca poprawnie scala dowolne dwa niemalejące ciągi zero-jedynkowe, to działa poprawnie dla ciągów niemalejących o dowolnych wartościach.
- 28.4-2. Ile różnych wejściowych ciągów zero-jedynkowych trzeba użyć, aby sprawdzić, czy sieć porównująca jest siecią scalającą.
- 28.4-3. Wykaż, że każda sieć scalająca jeden element z ciągiem $n - 1$ elementowym musi mieć głębokość co najmniej $\lg n$.
- * 28.4-4. Rozważ sieć scalającą z wejściami a_1, a_2, \dots, a_n , gdzie n jest dokładną potęgą 2, w której dwa ciągi rosnące przeznaczone do scalenia to $\langle a_1, a_3, \dots, a_{n-1} \rangle$ i $\langle a_2, a_4, \dots, a_n \rangle$. Udowodnij, że liczba komparatorów w tego typu sieci jest $\Omega(n \lg n)$. Dlaczego to dolne ograniczenie jest interesujące? (Wskazówka: Podziel komparatory na trzy zbiory).

28.5. SIEĆ SORTUJĄCA

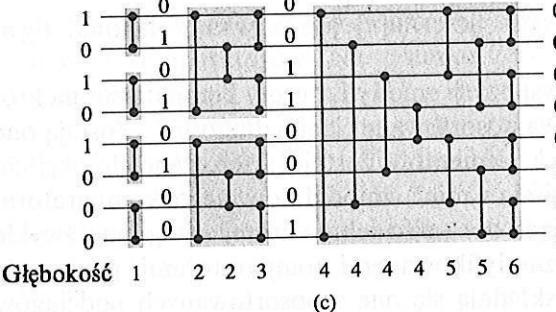
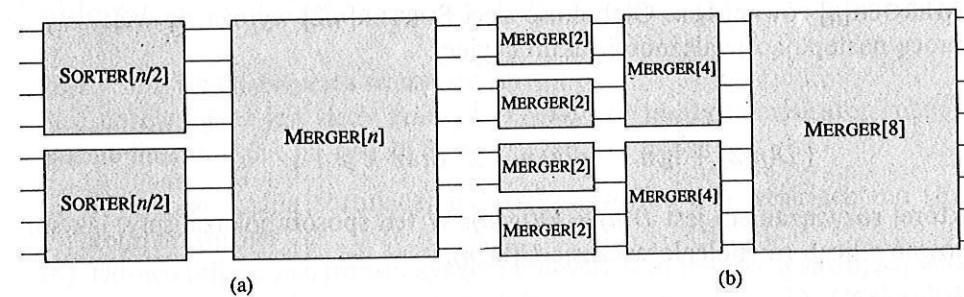
$\dots, a_{n-1} \rangle$ i $\langle a_2, a_4, \dots, a_n \rangle$. Udowodnij, że liczba komparatorów w tego typu sieci jest $\Omega(n \lg n)$. Dlaczego to dolne ograniczenie jest interesujące? (Wskazówka: Podziel komparatory na trzy zbiory).

- * 28.4-5. Udowodnij, że każda sieć scalająca, niezależnie od kolejności wejść, zawiera $\Omega(n \lg n)$ komparatorów.

28.5. Sieć sortująca

Zebralismy już wszystkie elementy wystarczające do skonstruowania sieci sortującej dowolny ciąg wartości. W skład sieci SORTER $[n]$ wchodzi sieć scalająca, będąca w istocie równoległym odpowiednikiem algorytmu sortowania przez scalanie z podrozdz. 1.3.1. Konstrukcja i działanie tej sieci są przedstawione na rys. 28.12.

Na rysunku 28.12a widać rekurencyjną strukturę sieci SORTER $[n]$. Ciąg n wartości na wejściu zostaje podzielony na połowy, które zostają posortowane rekurencyjnie (i równolegle) za pomocą dwóch kopii sieci SORTER $[n/2]$. Dwa posortowane ciągi są następnie scalone przez sieć MERGER $[n]$. Przypadkiem



Rys. 28.12. Sieć SORTER $[n]$ złożona rekurencyjnie z sieci scalających. (a) Konstrukcja rekurencyjna. (b) Rozwiniecie rekurencji. (c) Sytuacja po zastąpieniu pudełek scalających odpowiednimi sieciami scalającymi. Zaznaczona jest głębokość każdego komparatora oraz przykładowe wartości

granicznym dla rekursji jest sytuacja, w której $n = 1$, czyli trzeba posortować ciąg o długości 1. Ciąg taki jest oczywiście już posortowany. Na rysunku 28.12b widać wynik rozwinięcia rekursji, a na rys. 28.12c – całą sieć powstającą przez zastąpienie „pudełek” przez właściwe sieci scalające.

Dane przepływają przez $\lg n$ poziomów sieci $SORTER[n]$. Na każdym z wejść sieci jest już posortowany ciąg o długości 1. Pierwszy poziom sieci $SORTER[n]$ składa się z $n/2$ kopii sieci $MERGER[2]$, które równolegle scalają 1-elementowe ciągi, wyznaczając posortowane ciągi o długości 2. Na drugim poziomie, składającym się z $n/4$ kopii sieci $MERGER[4]$, są równolegle scalane 2-elementowe ciągi i wyznaczane posortowane ciągi o długości 4. Ogólnie, dla $k = 1, 2, \dots, \lg n$, poziom k składa się z $n/2^k$ kopii sieci $MERGER[2^k]$, które równolegle scalają 2^{k-1} -elementowe ciągi, wyznaczając posortowane ciągi o długości 2^k . Na ostatnim poziomie powstaje jeden posortowany ciąg, składający się z wartości wejściowych. Można przez indukcję wykazać, że ta sieć poprawnie sortuje ciągi zero-jedynkowe, więc przez odwołanie się do zasady zero-jedynkowej (twierdzenie 28.2) wiemy, że sortuje ciągi dowolnych wartości.

Głębokość powyższej sieci sortującej można wyrazić rekurencyjnie. Niech $D(n)$ oznacza głębokość sieci $SORTER[n]$. Jest ona równa sumie $D(n/2)$ (są dwie kopie sieci $SORTER[n/2]$, lecz działają one równolegle) oraz głębokości sieci $MERGER[n]$ równej $\lg n$. Głębokość sieci $SORTER[n/2]$ wyraża się więc za pomocą następującej zależności rekurencyjnej:

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

której rozwiązaniem jest $D(n) = \Theta(\lg^2 n)$. W ten sposób pokazaliśmy, jak sortować n liczb równolegle w czasie $O(\lg^2 n)$.

ZADANIA

28.5-1. Ile jest komparatorów w sieci $SORTER[n]$?

28.5-2. Wykaż, że głębokość sieci $SORTER[n]$ jest równa dokładnie $(\lg n)(\lg n + 1)/2$.

28.5-3. Założmy, że mamy do dyspozycji zmodyfikowany komparator, na którego wejście są podawane dwa posortowane ciągi o długości k . Zostają one scalone, po czym większe k elementów zostaje skierowane do wyjścia „max”, a pozostałe do wyjścia „min” zmodyfikowanego komparatora. Wykaż, że każda sieć sortująca o n wejściach, w której zastąpiono zwykłe komparatory powyższymi zmodyfikowanymi komparatorami, poprawnie sortuje ciągi nk liczb, jeśli składają się one z posortowanych podciągów o długości k .

28.5-4. Chcemy podzielić $2n$ elementów $\langle a_1, a_2, \dots, a_{2n} \rangle$ na dwie równe części tak, aby największe z nich znalazły się w jednej, a pozostałe w drugiej.

Wykaż, że po wcześniejszym posortowaniu ciągów $\langle a_1, a_2, \dots, a_n \rangle$ i $\langle a_{n+1}, a_{n+2}, \dots, a_{2n} \rangle$ można wykonać ten podział za pomocą sieci o stałej głębokości.

* **28.5-5.** Niech $S(k)$ będzie głębokością sieci sortującej o k wejściach, a $M(k)$ głębokością sieci scalającej o $2k$ wejściach. Przypuśćmy, że mamy ciąg n liczb do posortowania oraz wiemy, że każda liczba jest oddalona co najwyżej o k od swojej pozycji w ciągu posortowanym. Wykaż, że można posortować tych n liczb za pomocą sieci o głębokości $S(k) + 2M(k)$.

* **28.5-6.** Elementy macierzy $m \times m$ można posortować, powtarzając k razy następującą procedurę:

1. Posortuj każdy nieparzysty wiersz niemalejąco.
2. Posortuj każdy parzysty wiersz nierośnaco.
3. Posortuj każdą kolumnę niemalejąco.

Ile powtórzeń k potrzeba do posortowania tablicy oraz jak należy odczytywać posortowany ciąg?

Problemy

28-1. Sieci sortujące przez transpozycje

Sieć porównująca jest siecią transpozycyjną, jeśli każdy komparator łączy sąsiednie linie, tak jak na rys. 28.3.

- (a) Wykaż, że każda sortująca sieć transpozycyjna o n wejściach ma $\Omega(n^2)$ komparatorów.
- (b) Udowodnij, że sieć transpozycyjna o n wejściach jest siecią sortującą wtedy i tylko wtedy, gdy poprawnie sortuje ciąg $\langle n, n-1, \dots, 1 \rangle$. (Wskazówka: Użyj indukcji, podobnie jak w dowodzie lematu 28.1).

Sieć sortująca typu odd-even o n wejściach $\langle a_1, a_2, \dots, a_n \rangle$ składa się z n poziomów komparatorów. Na rysunku 28.13 znajduje się sieć transpozycyjna typu odd-even o 8 wejściach. Jak widać na rysunku, dla $i = 2, 3, \dots, n-1$ oraz $d = 1, 2, \dots, n$ linia i jest połączona za pomocą komparatora o głębokości d z linią $j = i + (-1)^{i+d}$, jeśli tylko $1 \leq j \leq n$.

- (c) Udowodnij, że rodzina sieci typu odd-even składa się rzeczywiście z sieci sortujących.

28-2. Sieć scalająca Batchera

W podrozdziale 28.4 opisaliśmy sieć scalającą opartą na sortowaniu bitonicznym. Zajmiemy się teraz konstrukcją sieci scalającej typu odd-even. Przymijmy,