Składowe spójności prostokątnych obrazów zero-jedynkowych

Łukasz Janiec

Politechnika Wrocławska
Wydział Elektroniki
Katedra Cybernetyki i Robotyki

17 grudnia 2020

Politechnika Wrocławska

Spis treści

Informacje o pracy

Temat

- **Składowe spójności prostokątnych obrazów zero-jedynkowych**

Wsparcie

- promotor: prof. dr hab. Krzysztof Bogdan (Politechnika Wrocławska)
- konsultant: dr Tomasz Hrycak (Austriacka Akademia Nauk)

Rodzaj pracy licencjackiej

- studialno-analityczna
- dziedzina matematyki: algorytmika, teoria grafów

Cel pracy

- Implementacja wydajnego algorytmu do wyznaczania i zliczania składowych spójności obrazów binarnych. Użyte środowisko - Matlab.
- Opis działania i analiza ww. algorytmu.

(*a*)

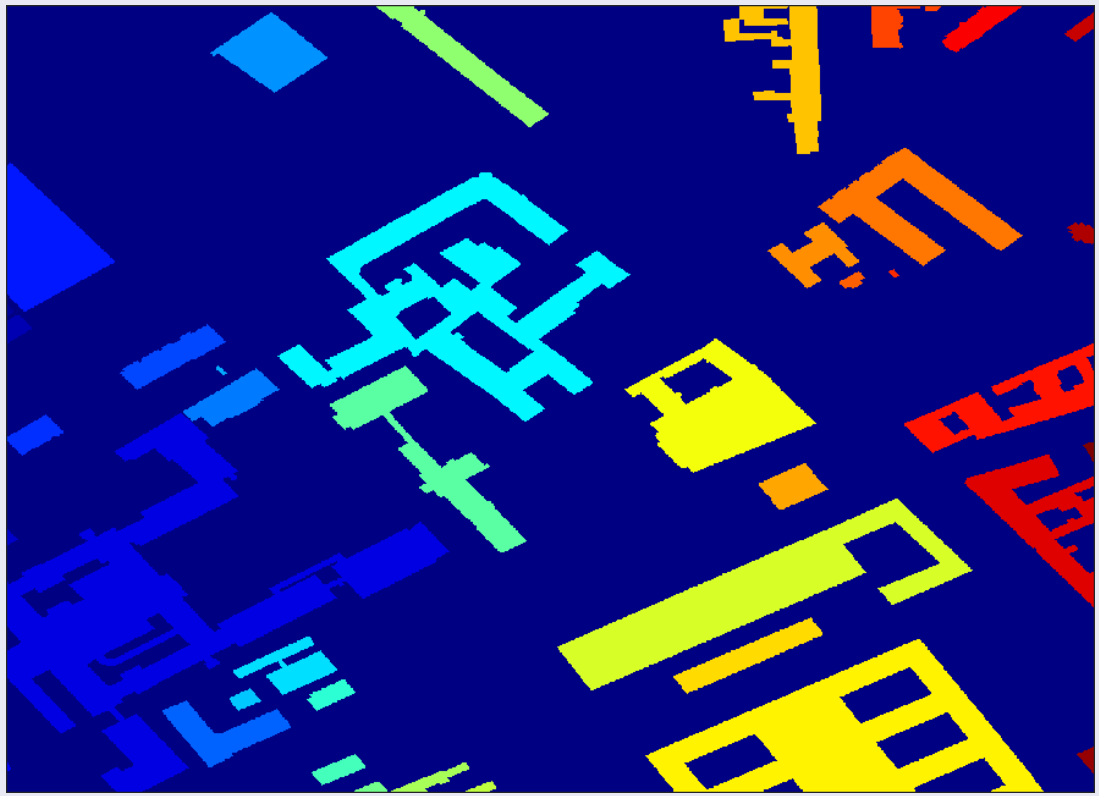
0	0	0	0	0	0	0
0	1	1	0	1	1	0
0	0	0	0	0	1	0
0	1	1	0	1	1	0
0	0	0	0	0	0	0

(*b*)

0	0	0	0	0	0	0
0	1	1	0	3	3	0
0	0	0	0	0	3	0
0	2	2	0	3	3	0
0	0	0	0	0	0	0

Rys.: (a) mapa binarna na wejściu, (b) składowe spójności na wyjściu

Duża mapa - efekt końcowy

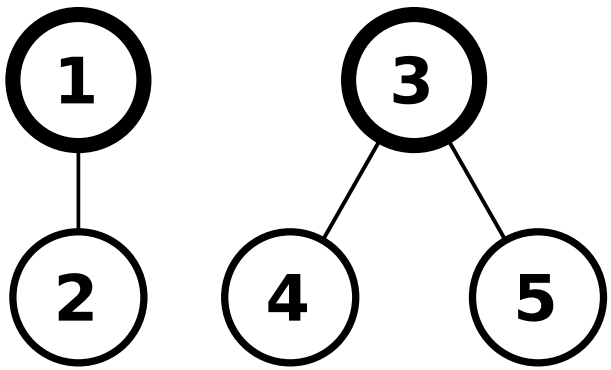


Przykład ze zbioru `street-map/Berlin_0_1024.map`.[1]
Wymiary mapy to 1024x1024 piksele. Czas przetwarzania na standardowym komputerze - poniżej 4s.

[1] Sturtevant, N., *Benchmarks for Grid-Based Pathfinding*

Spójny podzbiór obrazu jako składowa spójności

Załóżmy, że wyróżnione elementy obrazu oznaczono przez $1, 2, \ldots, N$. **Spójny podzbiór obrazu (składowa spójności)** modelujemy jako drzewo z wyróżnionym korzeniem, gdzie potomkowie przechowują numer rodzica, zaś w korzeniu umieszczamy wielkość drzewa pomnożoną przez (-1).

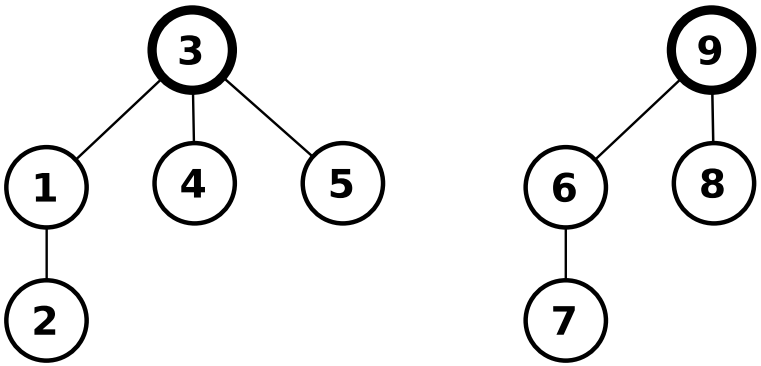


Składowe spójności jako lista adresowa

x	1	2	3	4	5
$V[x]$	**-2**	1	**-3**	3	3

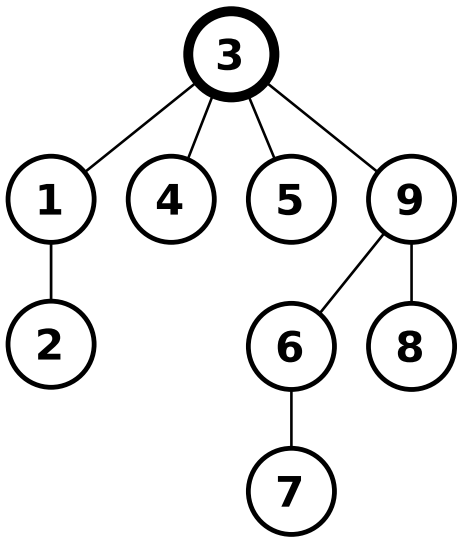
Taką strukturę danych nazywamy **strukturą zbiorów rozłącznych** (*ang.* union-find/disjoint-set structure).
Elementy obrazu w składowej powiązane są relacją sąsiedztwa 4-elementowego.

Podstawowe działanie - łączenie drzew



Przed połączeniem wierzchołków (3) i (9)

x	1	2	3	4	5	6	7	8	9
$V[x]$	3	1	**-5**	3	3	9	6	9	**-4**

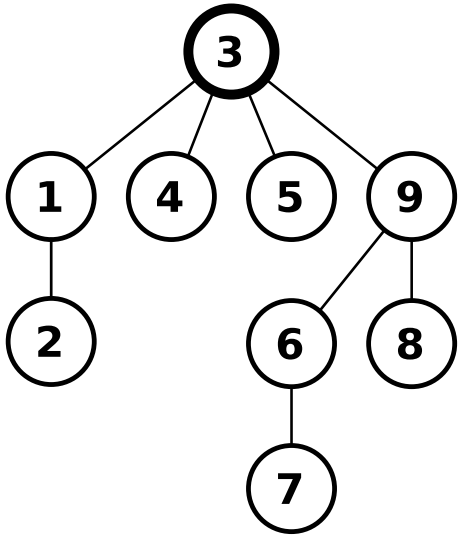


Po połączeniu (3) i (9)

x	1	2	3	4	5	6	7	8	9
$V[x]$	3	1	**-9**	3	3	9	6	9	3

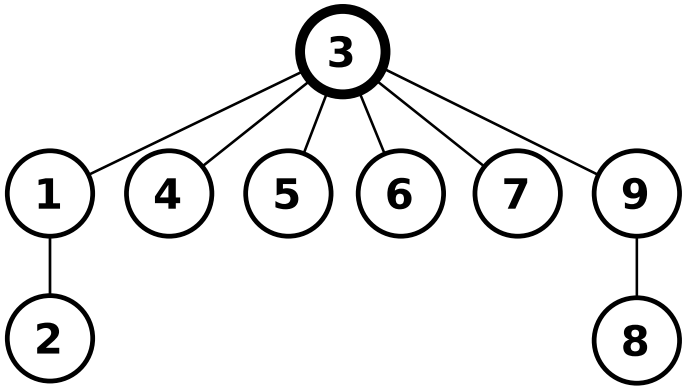
Przy łączeniu znane są tylko dwa łączone wierzchołki, dla których wyliczamy korzenie drzew, do których należą. Każde połączone drzewo jest na końcu kompresowane.

Podstawowe działanie - kompresja drzewa



Przed połączeniem wierzchołka (7) i korzenia (3)

x	1	2	3	4	5	6	7	8	9
$V[x]$	3	1	**-9**	3	3	9	6	9	3



Nowo powstałe skompresowane drzewo

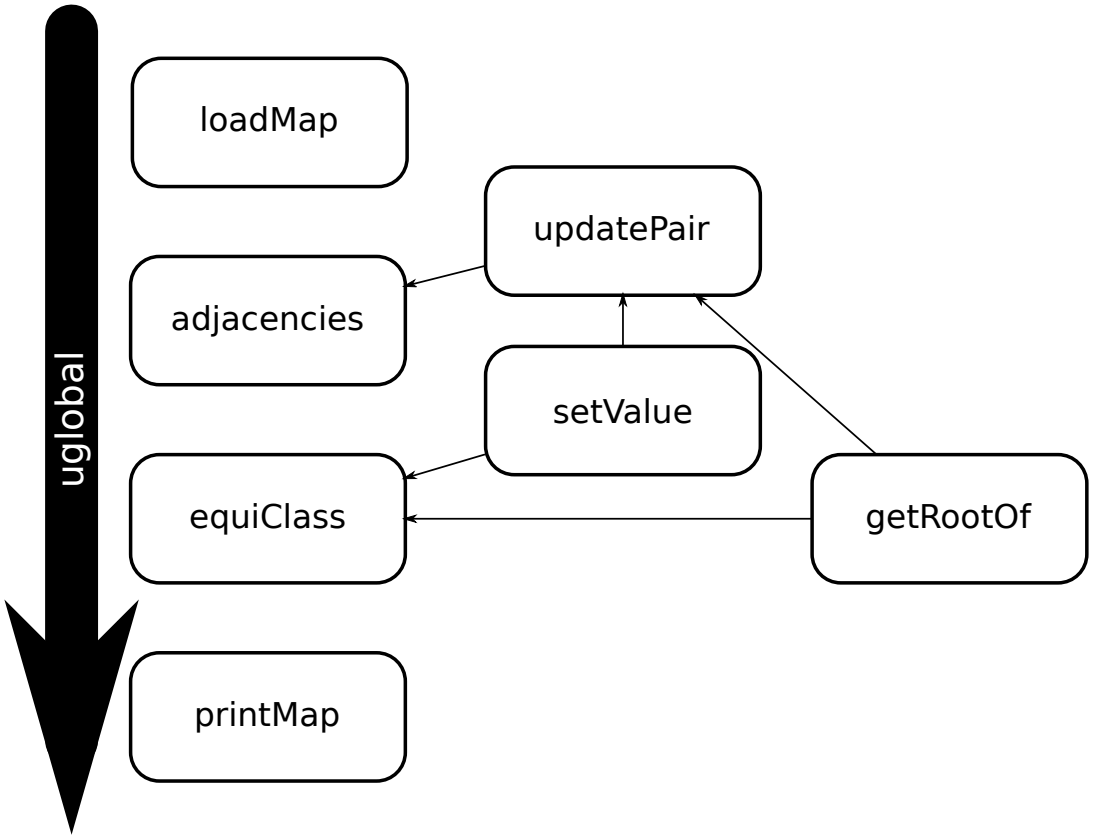
x	1	2	3	4	5	6	7	8	9
$V[x]$	3	1	**-9**	3	3	9	6	9	3

Kompresja odstępuje od konwencji abstrakcyjnych typów danych i rozdziału funkcji modyfikujących od funkcji dających dostęp do informacji o stanie obiektu, pozwalając na znaczny spadek złożoności obliczeniowej.

Przebieg algorytmu

1. Wczytaj mapę zero-jedynkową.
2. Zainicjuj dla każdej jedynki w obrazie osobną składową spójności z jednym wierzchołkiem (korzeniem).
3. Przetwarzaj piksele obrazu odpowiednio wzdłuż kolumn i wzdłuż wierszy. Dla każdej wykrytej relacji sąsiedztwa między jedynkami wywołaj połączenie drzew (jeśli drzewa są rozłączne) albo kompresję drzewa (gdy piksele były w tym samym drzewie).
4. Po przetworzeniu obrazu, każde wynikowe drzewo zawiera wszystkie jedynki powiązane ze sobą relacją komunikacji. Przypisz składowym rosnące indeksy od 1.
5. Zwróć maksymalny przypisany indeks jako liczbę składowych spójności i wyświetl przetworzoną mapę.

Funkcje podstawowe



Łączenie składowych spójności wykonuje `updatePair`, która używa `setValue` do kompresji drzewa i `getRootOf` do sprawdzania korzeni. Mapa zero-jedynkowa obecna w `uglobal` po wywołaniu `adjacencies` zmienia się na macierz relacji przylegania. Na końcu `equiClass` zlicza składowe spójności i oznacza każdą z nich jej indeksem.

Złożoność obliczeniowa algorytmu

Funkcja Ackermana $A(m, n)$

...to funkcja dwóch zmiennych, zdefiniowana rekurencyjnie

$$A(0, n) = n + 1, \quad A(m + 1, 0) = A(m, 1),$$
$$A(m + 1, n + 1) = A(m, A(m + 1, n)), n, m \in \mathbb{Z}_+. \tag{1}$$

Funkcja Ackermana rośnie niewyobrażalnie szybko[1] - pierwsze wartości są pokazane w tabeli poniżej.

A(m,n)	n=0	n=1	n=2	n=3	n=4	n=5
m=0	1	2	3	4	5	6
m=1	2	3	4	5	6	7
m=2	3	5	7	9	11	13
m=3	5	13	29	61	125	253
m=4	13	65533	$2^{65536} - 3$	$2^{265536} - 3$	$A(3,2^{265536} - 3)$	A(3,A(4,4))
m=5	65533	A(4,65533)	A(4,A(4,65533))	A(4,A(5,2))	A(4,A(5,3))	A(4,A(5,4))

Rys.: Weiss Stewart, *CSci 335 Software Design and Analysis III*, Chapter 8

[1] Wikipedia - Wartość $A(4, 4)$ wynosi $2^{2^{2^{2^{16}}}} - 3$.

Złożoność obliczeniowa algorytmu

Odwrotna funkcja Ackermana $\alpha(n)$

...jest zdefiniowana jako

$$\alpha(n) = \min\{m | A(m, m) \geqslant n\}\,. \tag{2}$$

Odwrotna funkcja Ackermana rośnie bardzo wolno - dla praktycznych zastosowań, funkcja $\alpha(\cdot)$ nie przekracza 5. Pozwala to na traktowanie jej jako w przybliżeniu stałej.

Złożoność obliczeniowa algorytmu

Zdefiniujmy $S = W \cdot H$, gdzie W - szerokość mapy, H - wysokość mapy, tj. S - ilość wszystkich pikseli przetwarzanej mapy.

Twierdzenie 1

Liczba operacji przy wyznaczaniu składowych spójności w obrazach zero-jedynkowych z uwzględnieniem wielkości drzew przy łączeniu i ich kompresji wynosi $\mathcal{O}(S\alpha(S))$, gdzie $\alpha(\cdot)$ jest odwrotną funkcją Ackermanna.

Dowód Twierdzenia 1

Oparty został na pracy Kozen, Dexter C. *The Design and Analysis of Algorithms*, rozdział "Analysis of Union-Find" (Springer NY, 1992).

Podsumowanie i informacje dodatkowe 1/2

- Wyznaczanie składowych spójności obrazów zero-jedynkowych z użyciem drzewiastej struktury zbiorów spójności ma praktycznie liniową czasową złożoność obliczeniową.
- Przykłady użycia - grafika komputerowa, segmentacja obrazów medycznych (MRI, TK), szukanie skupisk obiektów astronomicznych.

Podsumowanie i informacje dodatkowe 2/2

- Przy zmianie relacji sąsiedztwa, podejście tego algorytmu można zaadaptować do wyszukiwania składowych silnie spójnych w grafach skierowanych i sprawdzania rozwiązywalności labiryntów.
- Przy bardziej złożonych relacjach wiążących elementy składowych, złożoność obliczeniowa może się zwiększyć!
- Przy prawidłowej implementacji w Matlabie ważna staje się kolumnowa reprezentacja macierzy w pamięci i kolejność przetwarzania jej wymiarów.

Bibliografia

- Kozen, Dexter C., *The Design and Analysis of Algorithms*, "Analysis of Union-Find". Springer NY, 1992.
- Weiss Stewart, *CSci 335 Software Design and Analysis III*, "Chapter 8 Disjoint Sets and the Union/Find Problem". Hunter College, 2019.
- Sundblad, Yngve, *The Ackermann function. a theoretical, computational, and formula manipulative study*, BIT Numerical Mathematics, 1971.
- Tarjan Robert E., van Leeuwen Jan, *Worst-case Analysis of Set Union Algorithms*, J. ACM 31, 1984.
- `mathworks.com/company/newsletters/articles/programming-patterns-maximizing-code-performance\-by-optimizing-memory-access.html`

Dziękuję za uwagę!