Metody algorytmiczne

WĘDRUJ I SPRAWDZAJ

„Bądź cierpliwy, szukaj, to w końcu znajdziesz”

Algorytmy zbudowane w oparciu o jej schemat wymagają często dokonania przeglądu całej struktury danych:

- za pomocą pojedynczej pętli (iteracji), np. dla tablicy jednowymiarowej lub listy wskaźnikowej;
- za pomocą pętli (iteracji) zagnieżdżonych, np. dla tablicy wielowymiarowej lub listy list wskaźnikowych;
- za pomocą rekurencji, np. dla struktur drzewiastych.

Przyjmijmy upraszczające założenie:

Dokonanie przeglądu struktury danych = zbudowanie ciągu zawierającego wszystkie obiekty składowe z tej struktury

Drzewa także można przeglądać <u>iteracyjnie</u> (za pomocą pętli)

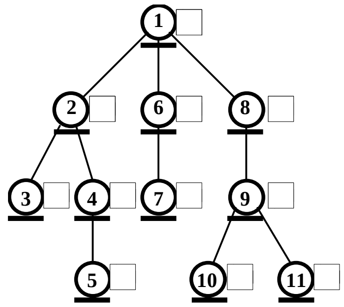
Dwa podstawowe schematy przeglądania drzewa ukorzenionego:

- przegląd drzewa w głąb,
- przegląd drzewa wszerz.

Algorytm przeglądu drzewa w głąb

(budowanie ciągu zawierającego wszystkie wierzchołki drzewa)

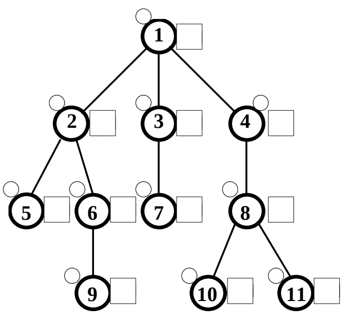
1. wstaw korzeń jako pierwszy element ciągu,
2. powtarzaj co następuje, aż do nadania korzeniowi etykiety „*zamknięty*”:
   2.1. wybierz z aktualnego ciągu <u>ostatni</u> wierzchołek, który <u>nie ma</u> etykiety „*zamknięty*”,
   2.2. jeśli wybrany wierzchołek nie ma potomstwa, które jeszcze nie zostało dopisane do ciągu, to nadaj mu etykietę „*zamknięty*”,
   2.3. w przeciwnym przypadku dopisz do ciągu <u>pierwszego</u> (licząc od lewej) jego potomka, który jeszcze nie występuje w ciągu.

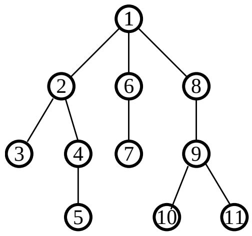


Algorytm przeglądu drzewa wszerz

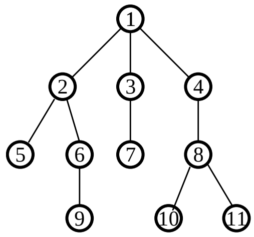
(budowanie ciągu zawierającego wszystkie wierzchołki drzewa)

1. nadaj wszystkim wierzchołkom drzewa etykietę „*nowy*”,
2. wstaw korzeń jako pierwszy element ciągu,
3. dopóki w tworzonym ciągu występuje wierzchołek z etykietą „nowy”, powtarzaj co następuje:
   3.1. wybierz z aktualnego ciągu <u>pierwszy</u> wierzchołek, który <u>ma</u> etykietę „*nowy*”,
   3.2. dodaj do ciągu <u>wszystkich</u> jego potomków w kolejności od lewej,
   3.3. usuń z wierzchołka wybranego w kroku 3.1. etykietę „*nowy*” .



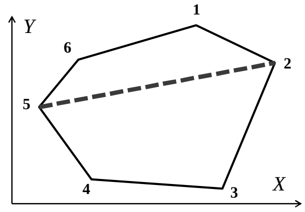


Przegląd drzewa w głąb



Przegląd drzewa wszerz

Algorytmy wyznaczania najdłuższej przekątnej w wielokącie wypukłym



N - liczba wierzchołków

Strukturą danych dla opisu wielokąta może być tablica dwuwymiarowa:

	1	2	3	...	*N*
1	x_1	x_2	x_3	...	x_N
2	y_1	y_2	y_3	...	y_N

Co trzeba przejrzeć, aby znaleźć najdłuższą przekątną?

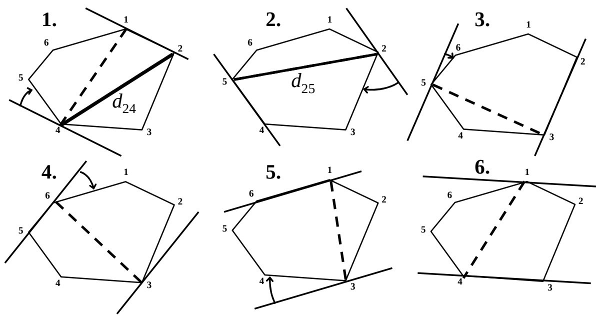
1. Algorytm „na siłę”:

1. Wypełnij tablicę dwuwymiarową *N*x*N* długościami wszystkich możliwych przekątnych,

	1	2	3	...	*N*
1	0	d_{12}	d_{13}	...	d_{1N}
2	d_{21}	0	d_{23}	...	d_{2N}
3	d_{31}	d_{32}	0	...	d_{3N}
...	...	...	...	...	...
N	d_{N1}	d_{N2}	d_{N3}	...	0

2. Przejrzyj $N(N-1)/2$ elementów w górnej trójkątnej połowie tablicy i znajdź największy.

2. Algorytm najlepszy ze znanych:



Dla N = **6**: 1. Algorytm – **15** kroków, 2. Algorytm – **6** kroków

DZIEL I ZWYCIĘŻAJ

Jeśli nie możesz uporać się od razu z rozwiązaniem zadania w całości, to spróbuj dzielić je na mniejsze o takiej samej strukturze i stosuj rekurencyjnie procedurę rozwiązywania.

Uzyskane rozwiązania małych fragmentów zadań łącz w rozwiązania tych zadań, które były wcześniej dzielone.

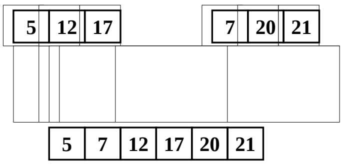
Przykładem zastosowania metody jest **algorytm sortowania przez scalanie**

Algorytm sortowania przez scalanie

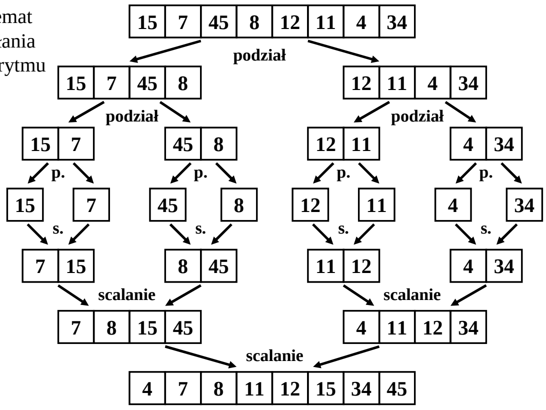
Dane wejściowe: nieuporządkowana lista (wskaźnikowa)

Procedura ***sortuj*** (*L*)
1. jeśli lista *L* zawiera tylko jeden element, to jest posortowana,
2. w przeciwnym przypadku wykonaj co następuje:
   2.1. podziel listę *L* na dwie połowy L_1 i L_2 ,
   2.2. wywołaj ***sortuj*** (L_1) ,
   2.3. wywołaj ***sortuj*** (L_2) ,
   2.4. scal posortowane listy L_1 i L_2 w jedną posortowaną listę ,
3. wróć do poziomu wywołania.

Schemat scalania w kroku 2.4:



Schemat działania algorytmu



METODA ZACHŁANNA

Bywają zadania, których rozwiązanie może być budowane z fragmentów dobieranych kolejno według zasady:
„idź naprzód, bierz najlepsze z tego co pod ręką i już nie oddawaj tego, co masz”.

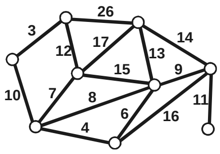
Przykładem zastosowania metody jest **algorytm wyznaczania minimalnego drzewa rozpinającego w grafie nieskierowanym**

Dane wejściowe:
skończona liczba punktów (*N*), odcinki łączące wybrane pary punktów (*M*) i liczby przypisane podanym odcinkom (ich długość, koszt, waga itp.)

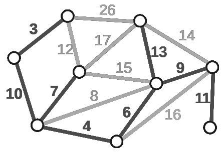
Wynik: taki zbiór odcinków, który tworzy najkrótszą sieć połączeń wiążącą wszystkie podane punkty

Algorytm zachłanny
do wyznaczania najkrótszej „sieci kolejowej”

Przykład dla N = 9 i M = 15:



1. Wybierz najkrótszy odcinek ze wszystkich podanych,
2. Powtarzaj co następuje, aż do połączenia wszystkich punktów:
   2.1. Wyznacz zbiór odcinków, które przyłączają do sieci jakiekolwiek jeszcze nie przyłączone miasto,
   2.2. Dołącz do sieci najkrótszy odcinek z tego zbioru



Długość sieci = **63**