

Obliczanie grafu widoczności

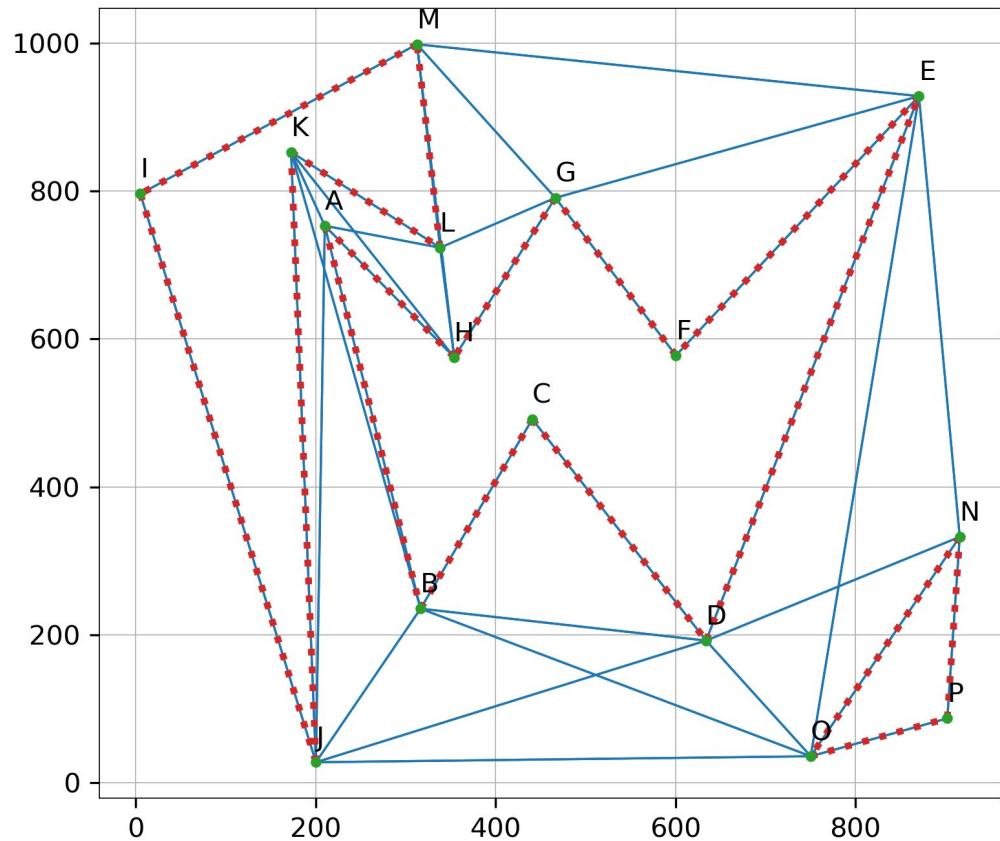


Algorytmy geometryczne
Projekt

Norbert Morawski
Dariusz Piwowarski

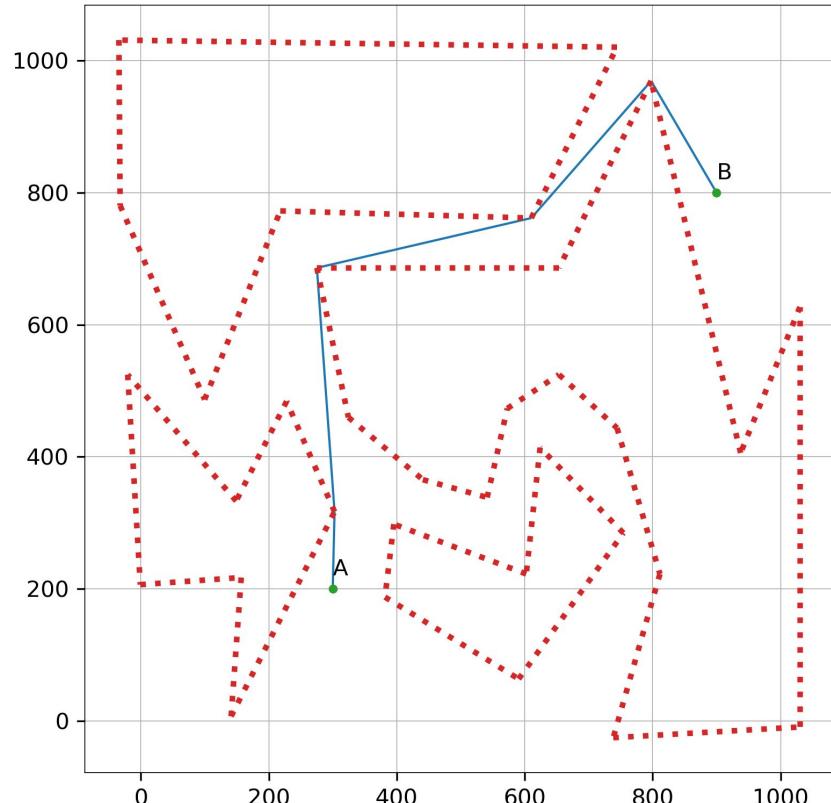
Graf widoczności

Jest to graf gdzie zbiorem wierzchołków jest zbiór punktów na płaszczyźnie, a krawędzie istnieją tylko kiedy wierzchołki "widzą się", tzn. nie istnieje pomiędzy nimi przeszkoda w postaci figury.



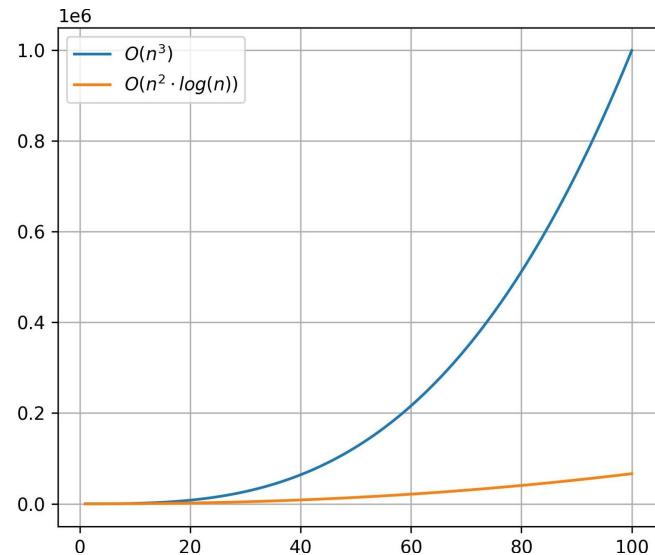
Zastosowanie

Jednym z zastosowań grafu widoczności jest możliwość wyznaczenia na jego podstawie najkrótszej ścieżki pomiędzy dwoma punktami na płaszczyźnie, na której znajdują się przeszkody w postaci wielokątów. Aby to zrobić wystarczy nadać krawędziom wagi odpowiadające odległościom pomiędzy punktami na płaszczyźnie, a następnie wykonać na takim grafie dowolny algorytm szukania najkrótszej ścieżki.



Wstęp

Trywialny algorytm wymagałby sprawdzenia każdej pary wierzchołków w czasie $O(n^2)$. Każde takie sprawdzenie wymagałoby przejrzenia wszystkich innych wierzchołków w czasie $O(n)$, co prowadziłoby do algorytmu o złożoności $O(n^3)$. Jednak jeżeli wykorzystamy zrównoważone drzewo binarne do wyszukiwania przeszkód, jesteśmy w stanie obniżyć złożoność do poziomu $O(n^2 \cdot \log(n))$.



Algorytm

Problem sprowadza się do wyznaczenia widzianych wierzchołków dla każdego z n punktów. Do tego celu wykorzystamy algorytm zamiatania. Płaszczyzna jest zamiatana poprzez obrót miotły zaczepionej w rozważanym punkcie, zgodny z ruchem wskazówek zegara.

Zdarzenia i stan miotły

Zdarzeniami będą **punkty na płaszczyźnie**, a struktura stanu będzie przechowywać **aktualnie przecinane odcinki** w kolejność od najbliższego punktowi zaczepienia do najdalszego.

Algorytm wyznaczania widocznych wierzchołków dla punktu p

Algorytm polega na zamiataniu płaszczyzny i odbywa się wg następujących kroków:

1. Posortuj wierzchołki na podstawie kąta (zgodnego z ruchem wskazówek zegara) jaki półprosta z p do danego wierzchołka tworzy z dodatnią półosią OX . Gdy kąt jest taki sam, wierzchołek bliższy p powinien znajdować się przed wierzchołkiem dalszym. Wierzchołki w tej kolejności umieść w strukturze zdarzeń.
2. Zaczep miotłę w badanym punkcie i zwróć ją w kierunku dodatniej półosi OX .
3. Dodaj do struktury stanu krawędzie przecinające miotłę w położeniu początkowym.

4. Dla kolejnych punktów w z struktury zdarzeń:

- Ustaw miotłę jako półprostą z **p** do **w**.
- Sprawdź czy punkt **w** jest widoczny, jeśli tak dodaj odpowiednią krawędź do grafu widoczności.
- Zaktualizuj strukturę stanu dodając do niej odcinki incydentne leżące po stronie zgodnej z ruchem wskazówek zegara względem miotły i usuwając te leżące po stronie przeciwnej do ruchu wskazówek zegara.

Struktura zdarzeń

Struktura zdarzeń została zaimplementowana przy użyciu listy.

Opis sortowania Zbiór punktów najpierw dzielony jest na dwa podzbiory względem tego czy dany punkt leży poniżej czy powyżej aktualnie rozpatrywanego punktu **p** (w przypadku gdy leżą one na tej samej wysokości co punkt p podział następuje po wartości współrzędnej **x**). Następnie każda część jest sortowana osobno wbudowaną funkcją sort (sortowanie stabilne) dwukrotnie.

W pierwszym sortowaniu kluczem jest odległość od punktu zaczepienia, w drugim stosowany jest wyznacznik aby posortować punkty zgodnie z ruchem wskazówek zegara.

Ostatecznie otrzymujemy posortowaną strukturę zdarzeń gdzie punkty występują w kolejności **zgodnie z ruchem wskazówek zegara**, a punkty wspólniowe ułożone są od **najbliższych do najdalszych**.

Złożoność Struktura zdarzeń nie jest modyfikowana w trakcie działania algorytmu, stąd jej wkład do złożoności jest ograniczony przez sortowanie i wynosi $O(n \cdot \log(n))$.

Struktura stanu

Struktura stanu przechowuje krawędzie. Wykorzystuje posortowaną listę (opartą na drzewie binarnym) **SortedList** z biblioteki sortedcontainers. Gwarantuje ona czas operacji rzędu $O(\log(n))$.

Opis działania Gdy miotła natrafia na punkt, brane są pod uwagę krawędzie incydentne z nim. Jeżeli krawędź została już zamieciona, jest ona usuwana ze struktury. Gdy nie została jeszcze zamieciona, jest dodawana do niej. Test po której stronie znajduje się krawędź jest przeprowadzany przy użyciu wyznacznika.

Porządek elementów Elementy są utrzymywane w porządku rosnącym względem ich odległości od punktu zaczepienia miotły. Jeżeli oba odcinki zaczynają lub kończą się w tym samym punkcie używamy wyznacznika.

Złożoność Na strukturze stanu wykonamy maksymalnie wykonamy n operacji wstawiania i usuwania, co daje nam czas działania rzędu $O(n \cdot \log(n))$.

Sprawdzanie widoczności

Zgodnie z opisem algorytmu sprawdzanie widoczności będziemy wykonywać dla zadanych punktów **p** i **w**. W celu określenia, czy punkt **w** jest widoczny będziemy analizować strukturę stanu miotły. W niektórych przypadkach potrzebujemy dodatkowe informacje o punkcie z struktury zdarzeń, który był rozważany poprzednio. Oznaczamy ten punkt jako **prev_w**.

Gdy zaczynamy rozpatrywać punkt, testujemy czy jest on widziany z punktu zaczepienia miotły. W algorytmie spotykamy następujące przypadki:

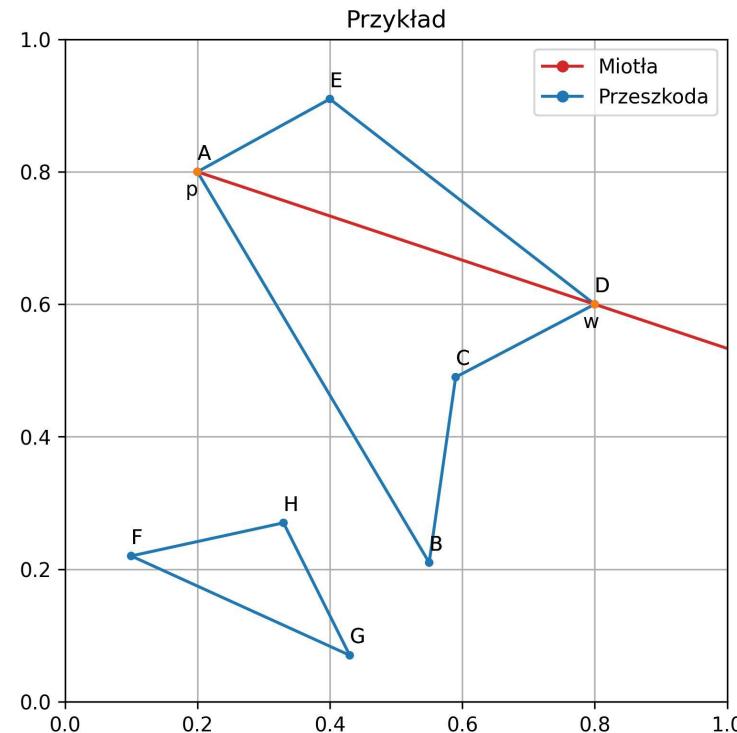
Przypadek 1

Punkty **p** i **w** należą do tej samej figury i wektor **wp** jest skierowany do wnętrza wielokąta.

W tym przypadku punkt **w** nie jest widoczny, ponieważ linia łącząca **p** i **w** przechodziłaby przez wnętrze figury.

Przypadek musimy sprawdzić jako pierwszy, ponieważ taka sytuacja spełnia również przypadek 2 lub 3 i wtedy punkt **w** zostałby błędnie oznaczony jako widoczny.

Jak widać na przykładzie wektor **wp** jest skierowany do wnętrza wielokąta, zatem pomimo że miotła nie jest przecinana przez żaden odcinek punkt **w** nie jest widoczny.

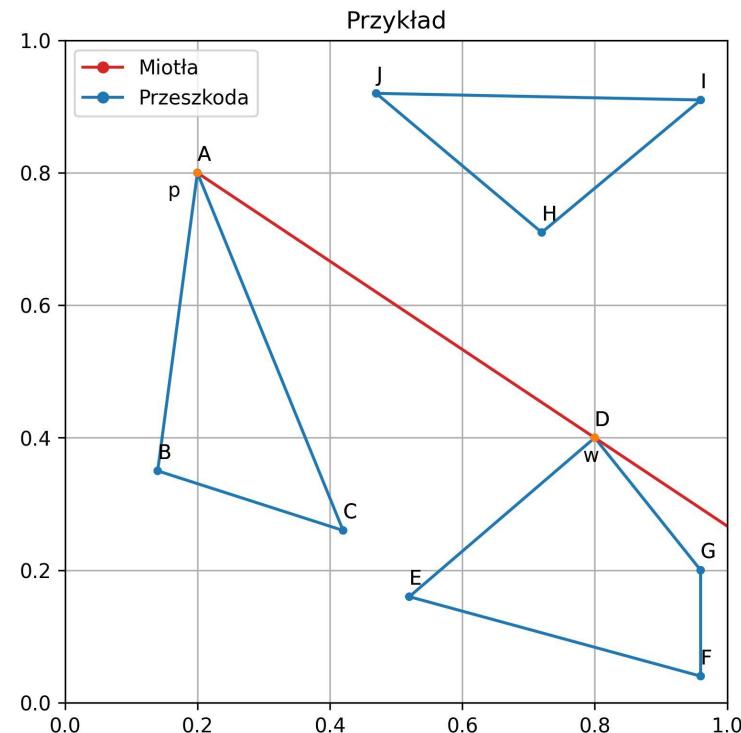


Przypadek 2

Nie zaszedł przypadek 1 i struktura stanu miotły jest pusta.

Struktura stanu jest pusta zatem na linii z p do w nie może pojawić się żadna przeszkoda.

Punkt w jest zatem widoczny.



Przypadek 3

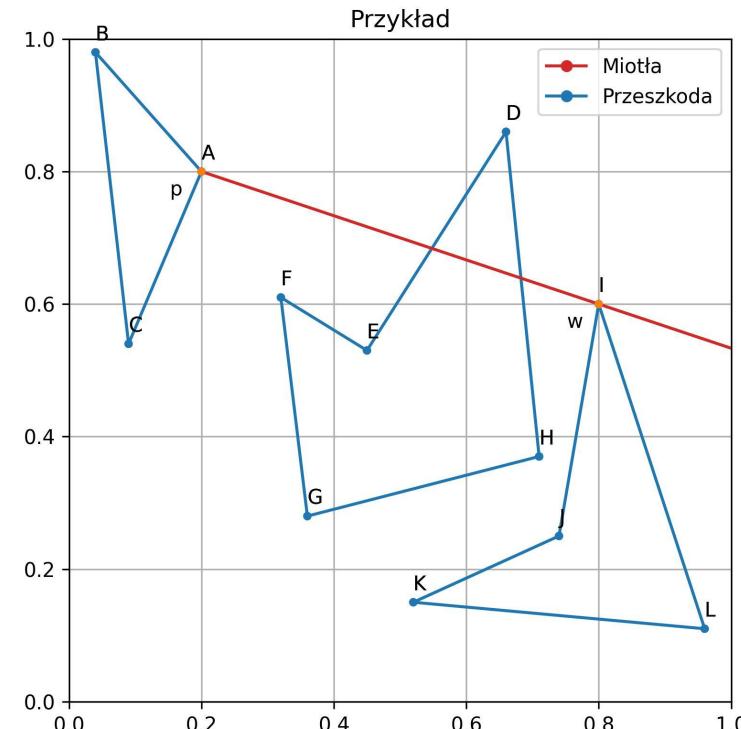
Nie zaszedł żaden z powyższych przypadków i punkty **p**, **prev_w**, **w** nie są współliniowe.

Jest to przypadek, który standardowo będzie występował najczęściej. Aby stwierdzić czy punkt **w** jest widoczny należy sprawdzić, czy punkt przecięcia odcinka znajdującego się na samym początku struktury stanu (czyli najbliżej **p**) z miotłą leży bliżej, czy dalej od **p** niż **w**.

Inaczej mówiąc sprawdzamy, czy odcinek **pw** przecina odcinek z struktury stanu. Jeżeli tak jest to oznacza, że pomiędzy **p** i **w** znajduje się przeszkoda i punkt **w** jest niewidoczny.

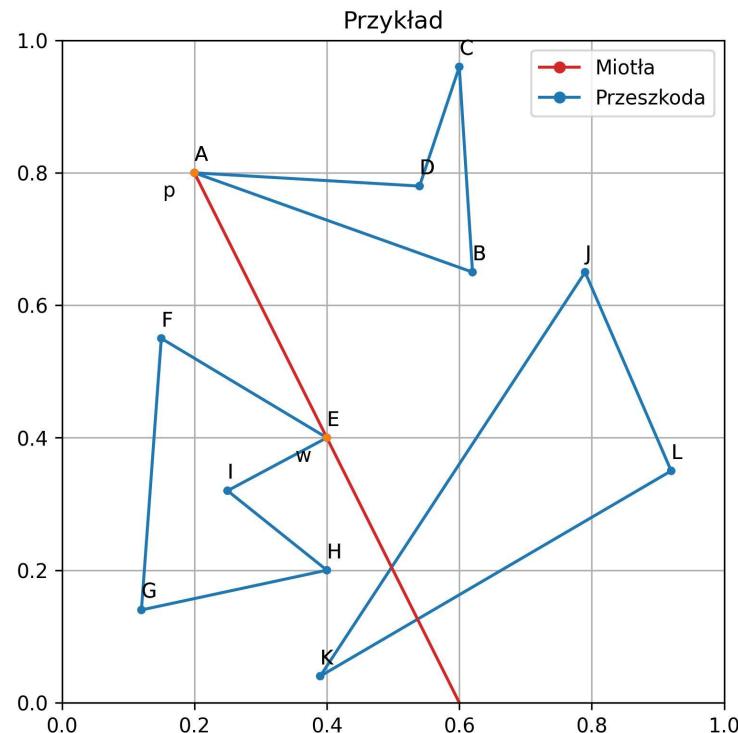
W przeciwnym wypadku punkt ten jest widoczny.

Pierwszy odcinek z struktury stanu to **DE**, przecina się on z odcinkiem **pw**, zatem punkt **w** nie jest widoczny.



Przypadek 3

Pierwszy odcinek z struktury stanu to **KJ**, nie przecina się on z odcinkiem **pw**, zatem punkt **w** jest widoczny.

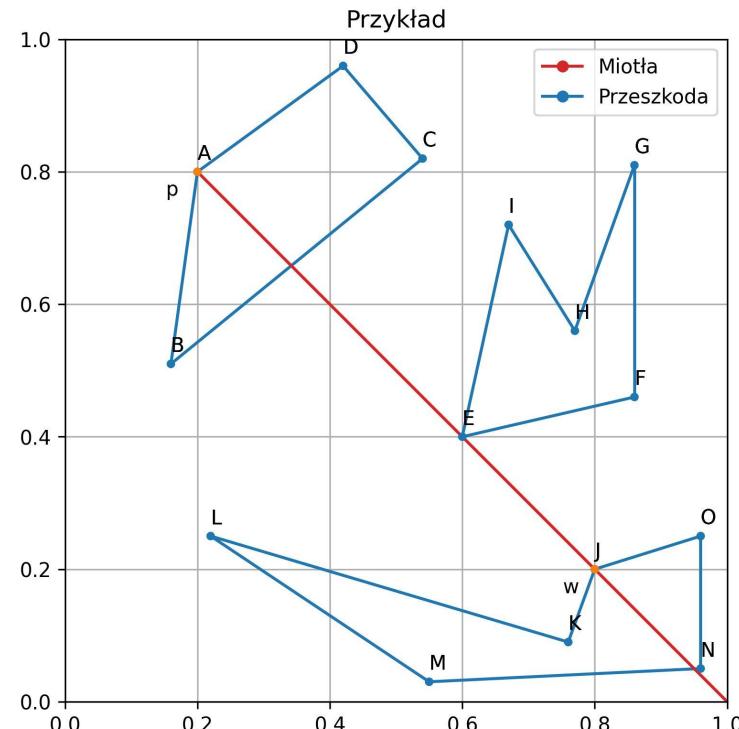


Przypadek 4

Nie zaszedł żaden z powyższych przypadków, punkty **p**, **prev_w**, **w** są współliniowe oraz punkt **prev_w** nie był widoczny:

Współliniowe punkty w strukturze zdarzeń, są ustawione w kolejności rosnącej odległości od **p**, zatem jeżeli **prev_w** był niewidoczny, to **w** także nie jest widoczny.

Punkt **E** był rozważany bezpośrednio przed punktem **w** i jest on niewidoczny. Dodatkowo punkty **p**, **E**, **w** są współliniowe. Wynika z tego, że punkt **w** również nie jest widoczny.



Przypadek 5

Nie zaszedł żaden z powyższych przypadków, punkty **p**, **prev_w**, **w** są wspólniowe oraz punkt **prev_w** był widoczny:

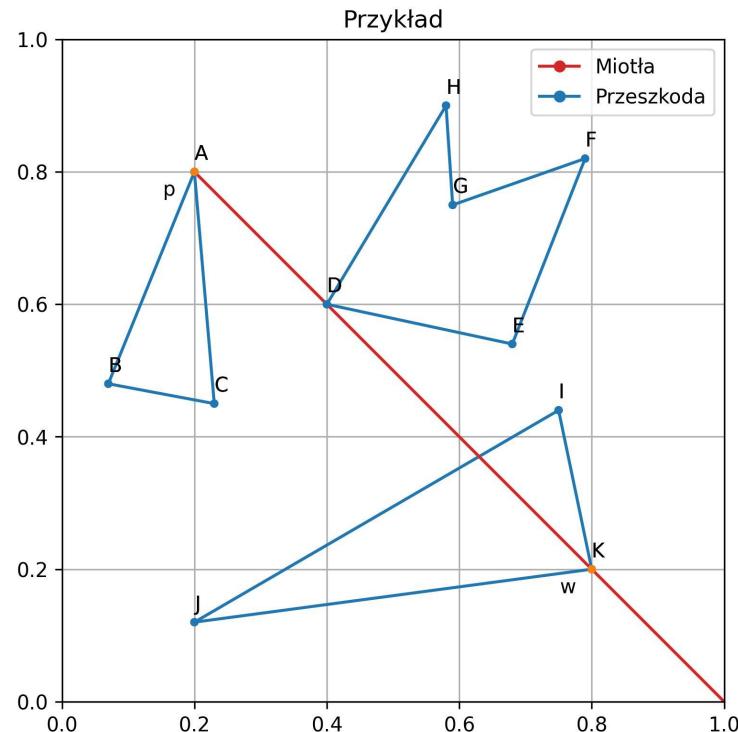
Przypadek ten jest podobny do przypadku 3, jednak musimy rozważyć go osobno, ponieważ po przetworzeniu punktu **prev_w** na początku struktury stanu mogły zostać umieszczone odcinki, które nie przecinają miotły. Aby w tym przypadku stwierdzić, czy punkt **w** jest widoczny sprawdzamy, czy w strukturze stanu znajduje się odcinek, który przecina **w-prev_w**.

Gdy nie znajdziemy w strukturze stanu takiego odcinka, dodatkowo musimy rozważyć jeszcze jeden szczególny pod-przypadek, podobny do przypadku 1. Zachodzi on, gdy **prev_w** i **w** należą do tej samej figury i wektor **w-prev_w** jest skierowany do wnętrza wielokąta.

Jeśli zachodzi którakolwiek z opisanych powyżej sytuacji, to **w** jest niewidoczny. W przeciwnym wypadku punkt ten jest widoczny.

Przypadek 5

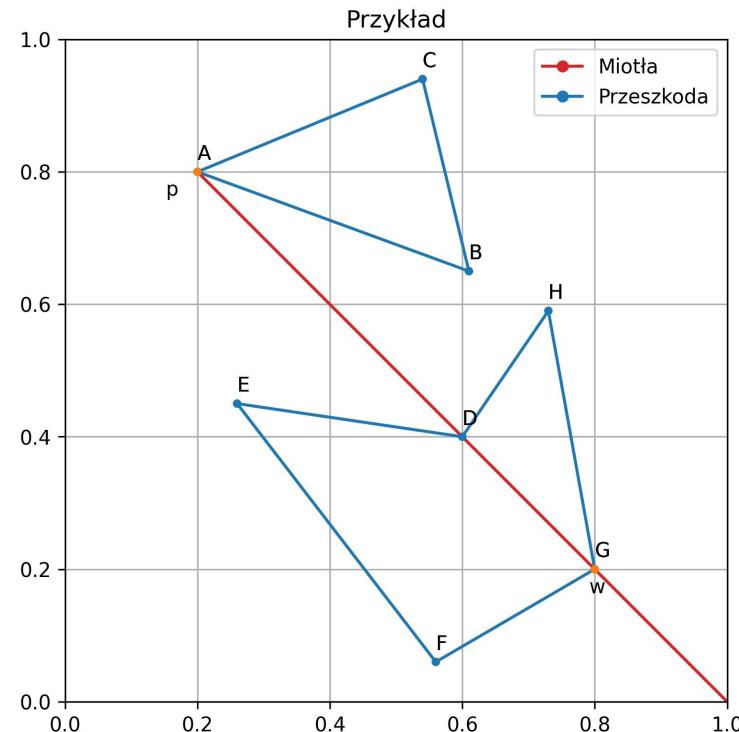
Punkt **D** był rozważany bezpośrednio przed punktem **w** i jest widoczny. Punkty **p**, **D**, **w** są współliniowe. W strukturze stanu znajduje się odcinek **IJ**, który przecina **Dw**. Wynika z tego, że **w** jest niewidoczny.



Przypadek 5

Punkt **D** był rozważany bezpośrednio przed **w** i jest widoczny, **p**, **D**, **w** są współliniowe.

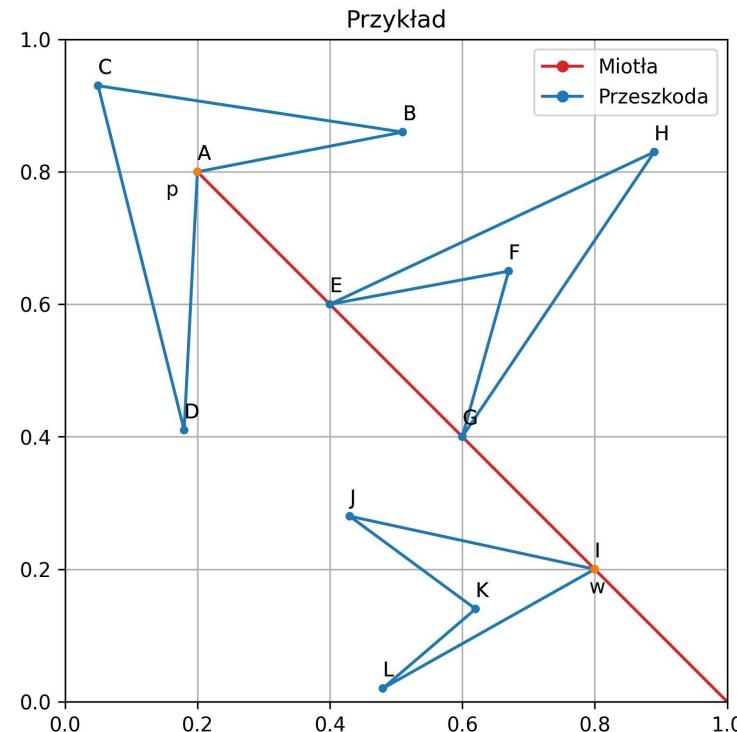
W strukturze stanu nie znajduje się odcinek, który przecina **Dw**, ale **w** i **D** należą do tej samej figury i wektor **wD** jest skierowany do wnętrza wielokąta, zatem **w** jest niewidoczny.



Przypadek 5

Punkt **G** był rozważany bezpośrednio przed punktem **w** i jest widoczny. Punkty **p**, **G**, **w** są współliniowe.

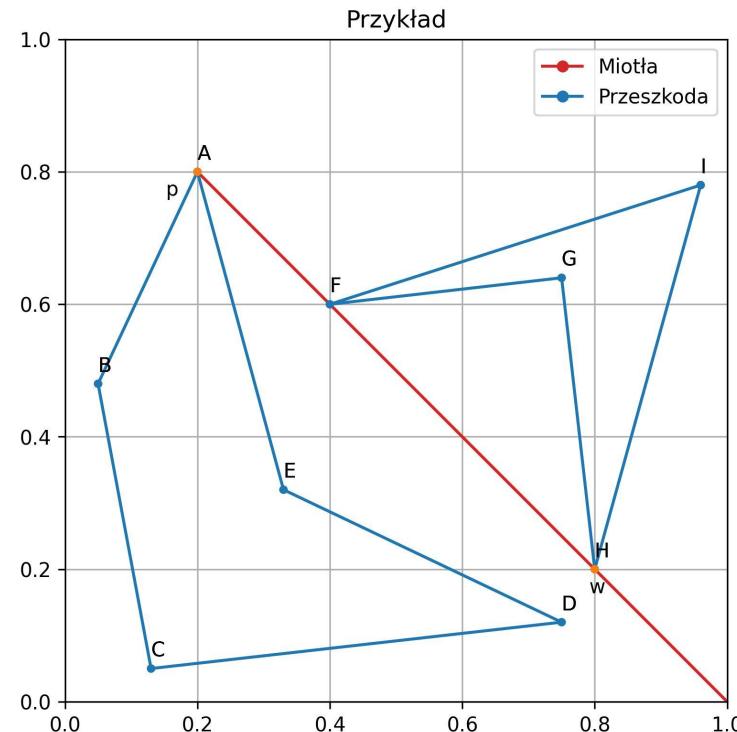
W strukturze stanu nie znajduje się odcinek, który przecina **Gw**. Punkty **w** i **G** należą do różnych figur. Wynika z tego, że **w** jest widoczny.



Przypadek 5

Punkt **F** był rozważany bezpośrednio przed punktem **w** i jest widoczny. Punkty **p**, **F**, **w** są współliniowe.

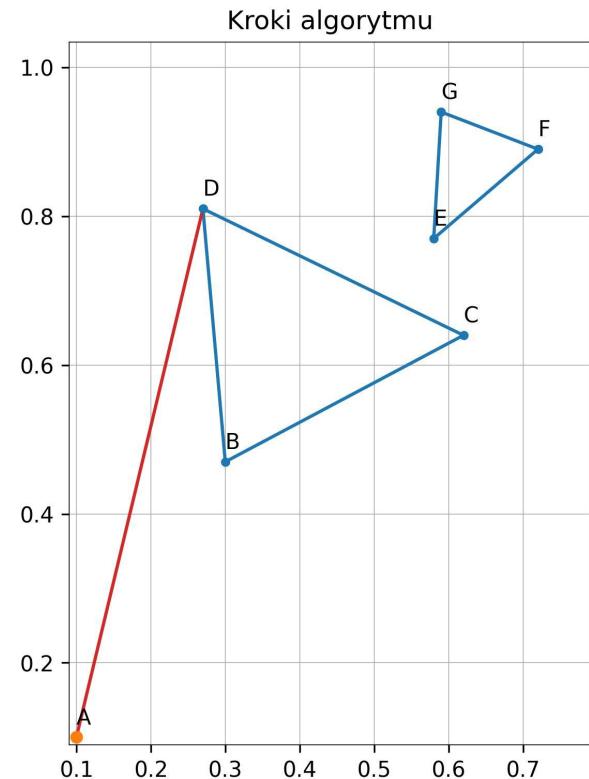
W strukturze stanu nie znajduje się odcinek, który przecina **Fw**. Punkty **w** i **F** należą do tej samej figury, ale wektor **wF** nie jest skierowany do wnętrza wielokąta, zatem **w** jest widoczny.



Kroki działania algorytmu

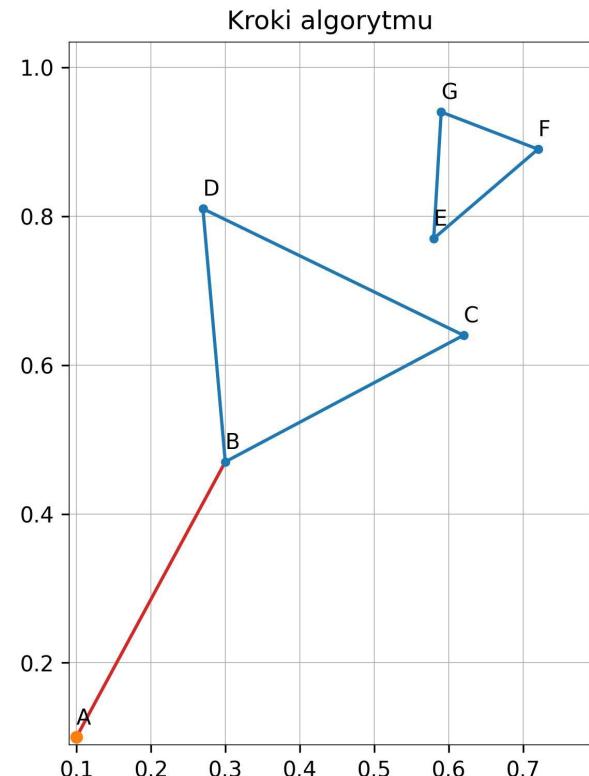
Punkt **A** jest punktem zaczepienia miotły. Miotła jest reprezentowana jako czerwony odcinek z **A** do obecnie przetwarzanego punktu.

Algorytm napotyka pierwsze zdarzenie. Punkt **D** jest widoczny, ponieważ struktura stanu jest pusta. Odcinki incydentne **DB** i **DC** dodawane są do struktury stanu (w tej kolejności).



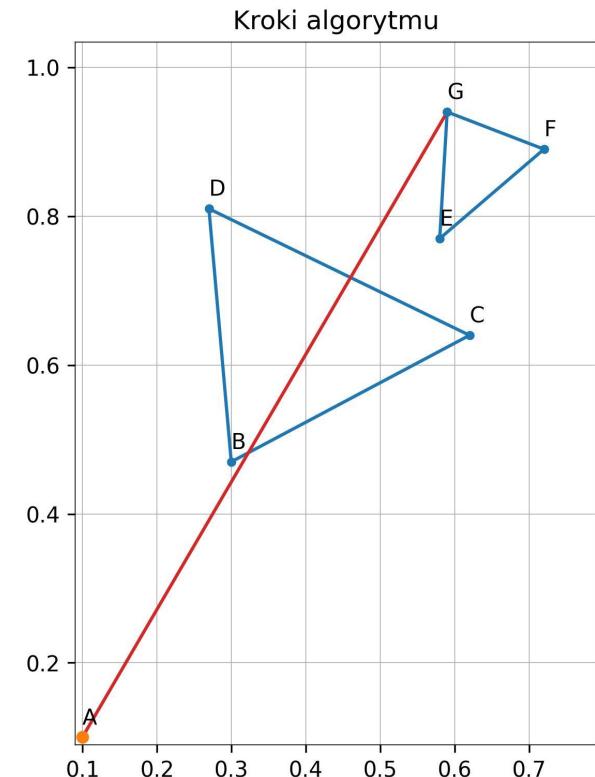
Kroki działania algorytmu

DB jest usuwane ze struktury, **BC** zostaje dodane.
Punkt **B** jest widoczny, ponieważ nie istnieje krawędź
przecinająca miotłę pomiędzy **A** i **B**.



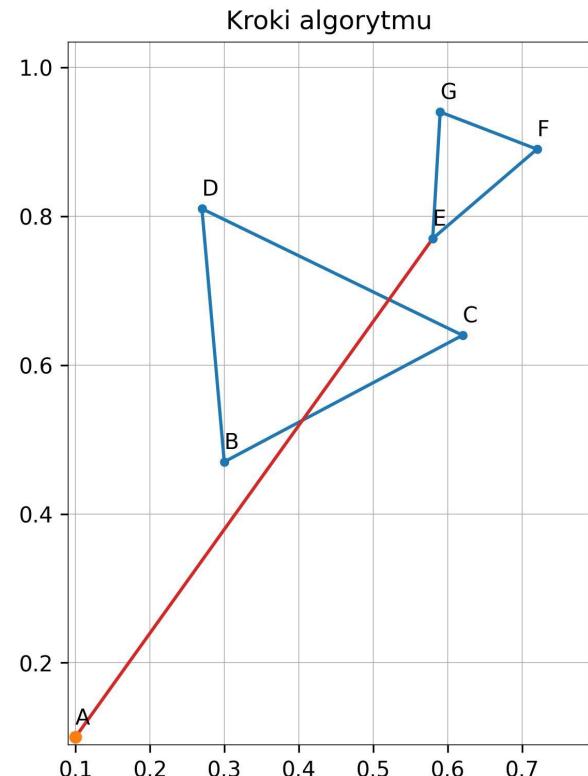
Kroki działania algorytmu

Dodajemy do struktury **GF** i **GE**. Punkt **G** nie jest widoczny, bo zasłania go krawędź **BC** (pierwsza przecinająca miotłę).



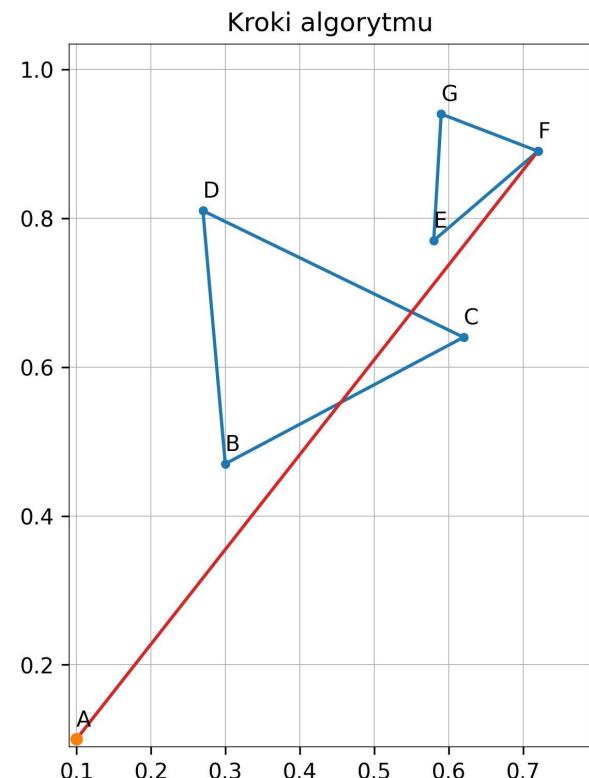
Kroki działania algorytmu

Usuwamy **GE**, dodajemy **EF**. Analogicznie do **G**, punkt **E** nie jest widoczny.



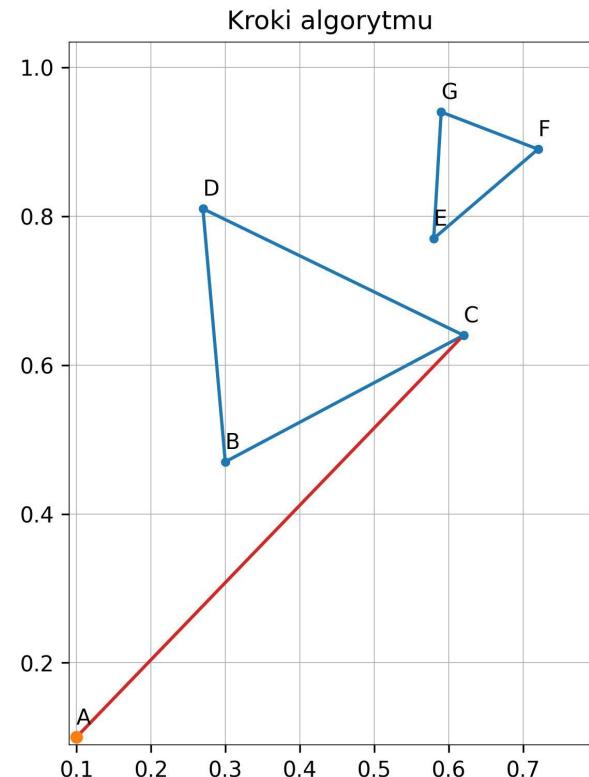
Kroki działania algorytmu

Usuwamy **EF** i **GF**. Punkt **F** niewidoczny.



Kroki działania algorytmu

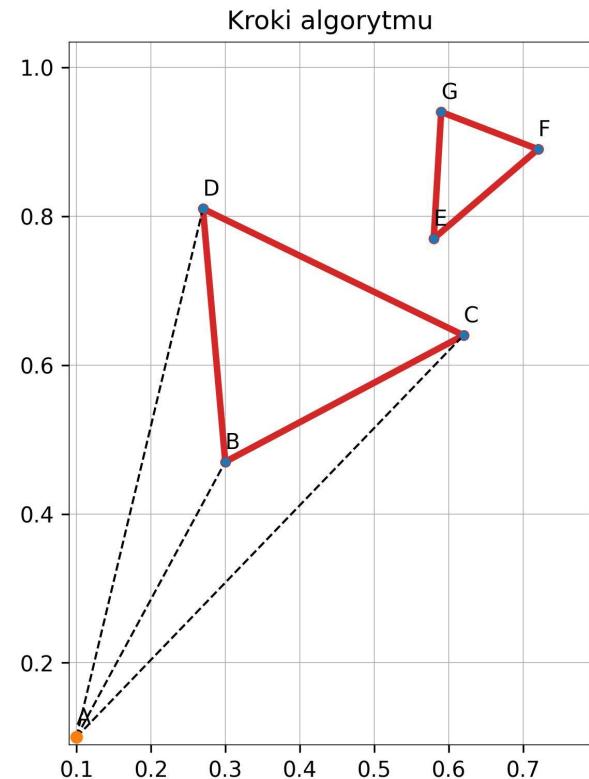
Usuwamy **BC** i **DC**. Punkt **C** widoczny.



Kroki działania algorytmu

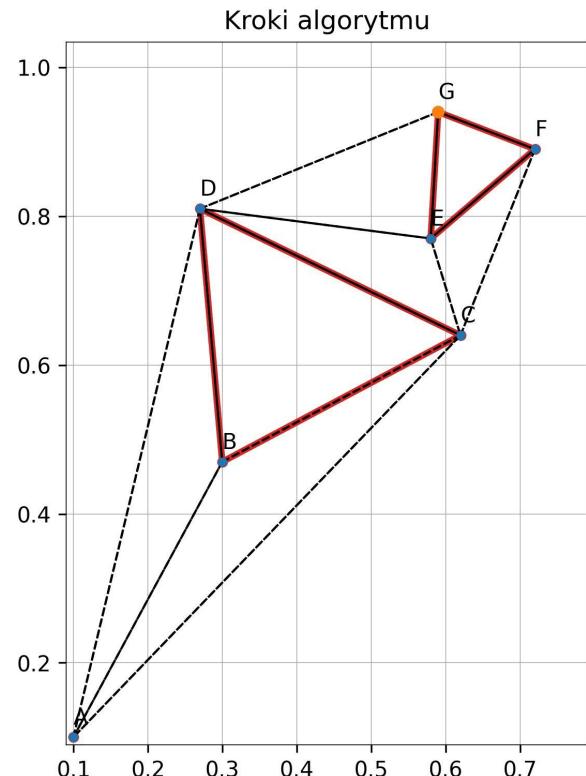
Tak wygląda wynik wykonania powyższych kroków. Następnie wybierany jest nowy punkt zaczepienia i algorytm powtarza kroki wykonane powyżej.

Kolorem czerwonym oznaczone są figury (przeszkody), krawędzie dodane do grafu widoczności zaznaczone zostały przerywanymi liniami.



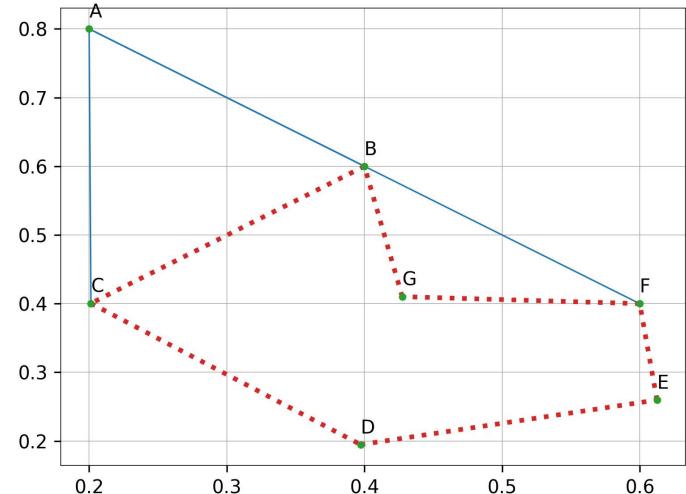
Kroki działania algorytmu

Tak wygląda ostatecznie graf widoczności dla tego przypadku.



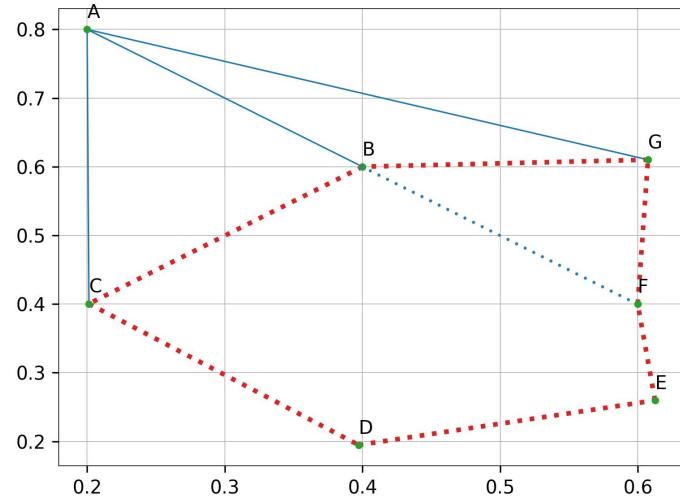
Testowanie działania algorytmu

Wierzchołek **F** jest widoczny, ponieważ **B** jest widoczny i miotły nie przecinają żadne krawędzie na odcinku **BF** oraz wektor **FB** nie jest skierowany do wnętrza figury.



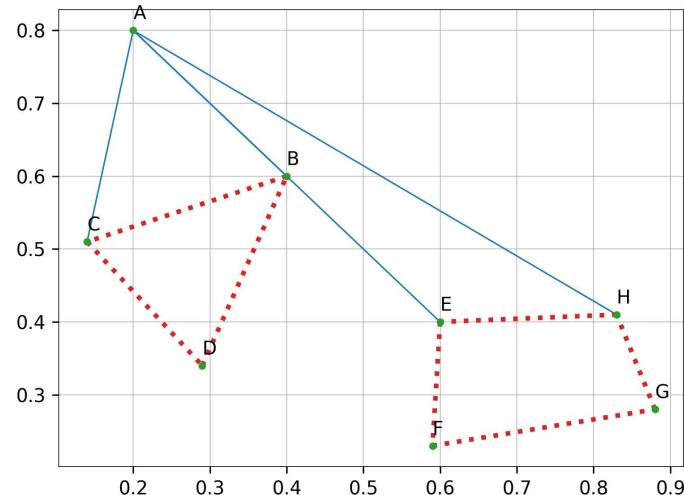
Testowanie działania algorytmu

Tutaj wektor **FB** jest skierowany do wnętrza figury więc punkt **F** jest niewidoczny (przerywana linia ma za zadanie tylko podkreślić fakt, że punkty **B** i **F** są współliniowe).



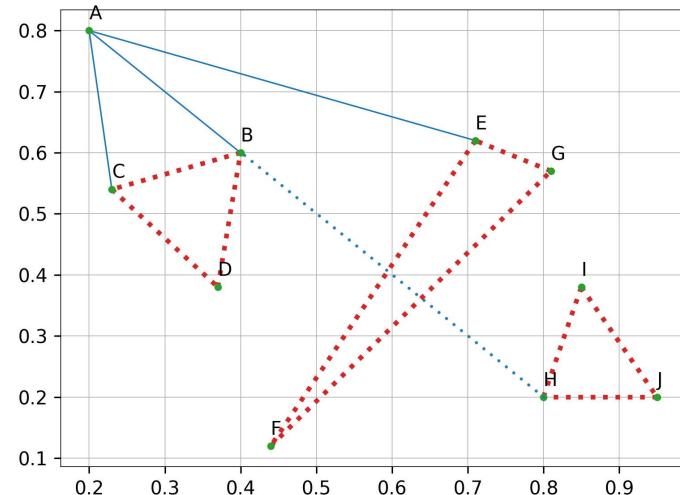
Testowanie działania algorytmu

Punkt **E** jest widoczny, ponieważ **B** jest widoczny i należą do różnych figur oraz pomiędzy nimi nie ma żadnej krawędzi przecinającej odcinek **BE**.



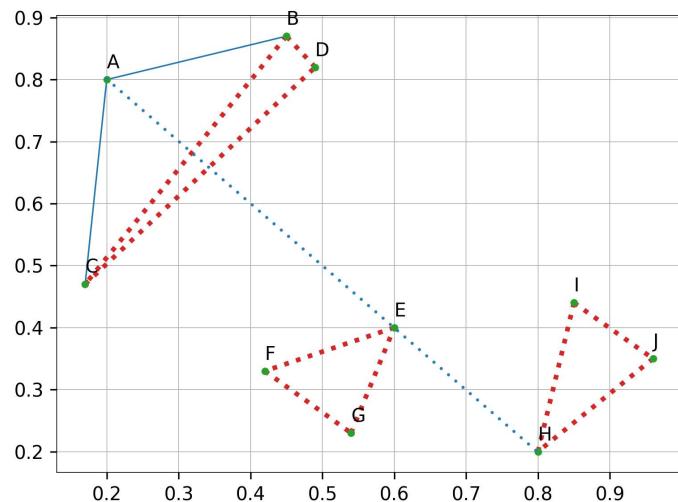
Testowanie działania algorytmu

Tutaj analogicznie jak wcześniej, **B** jest widoczny i punkty **B** i **H** należą do różnych figur, ale **BH** jest przecięty dwoma odcinkami więc **H** nie będzie widoczny.



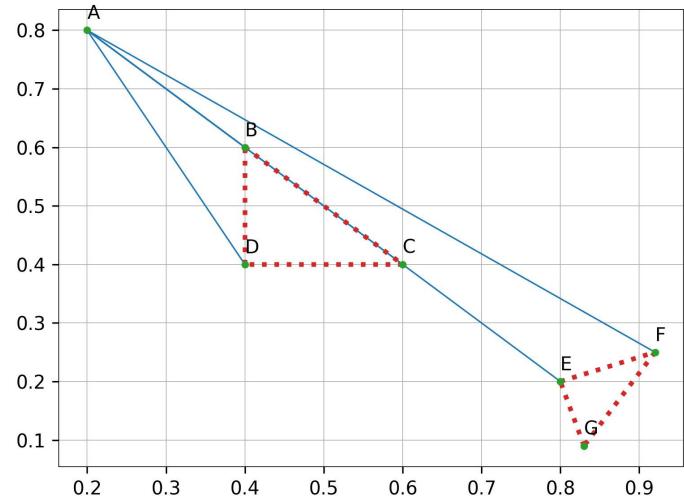
Testowanie działania algorytmu

Jeżeli **E** jest niewidoczny to nie ma sensu badać widoczności **H**, ponieważ jest on dalej od punktu zaczepienia miotły (punktu **A**).

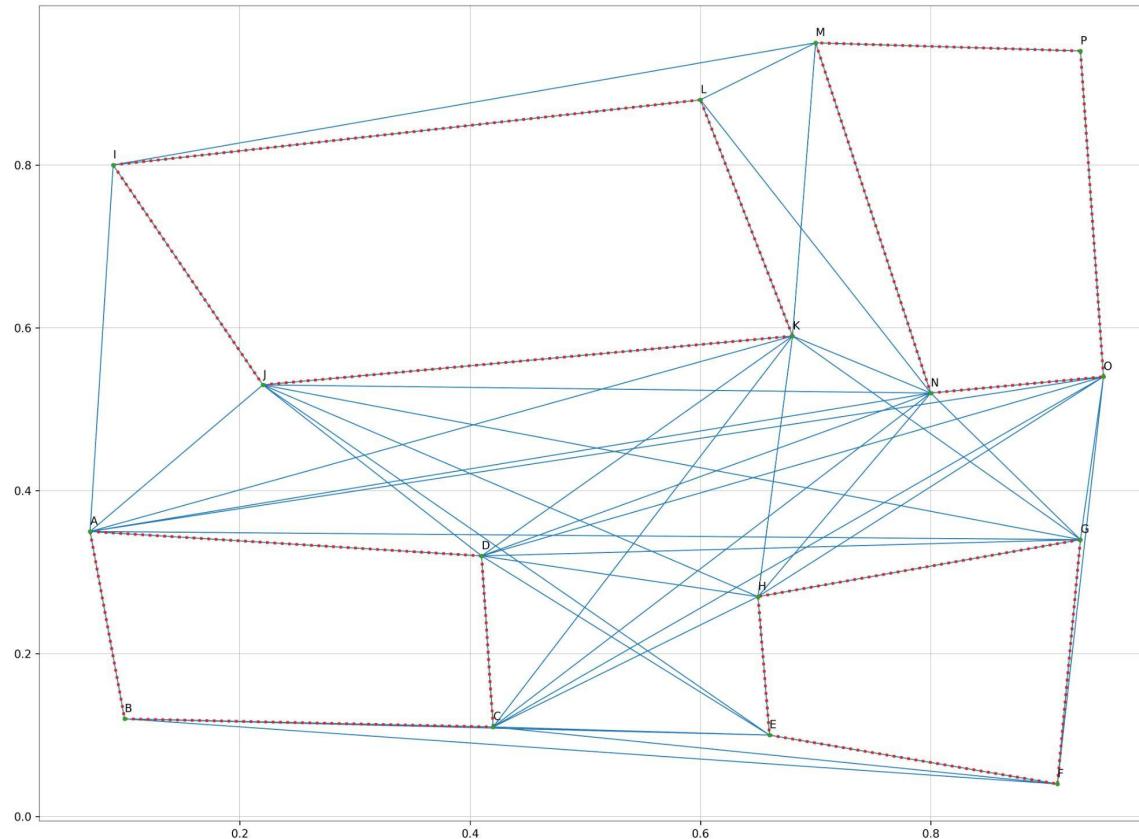


Testowanie działania algorytmu

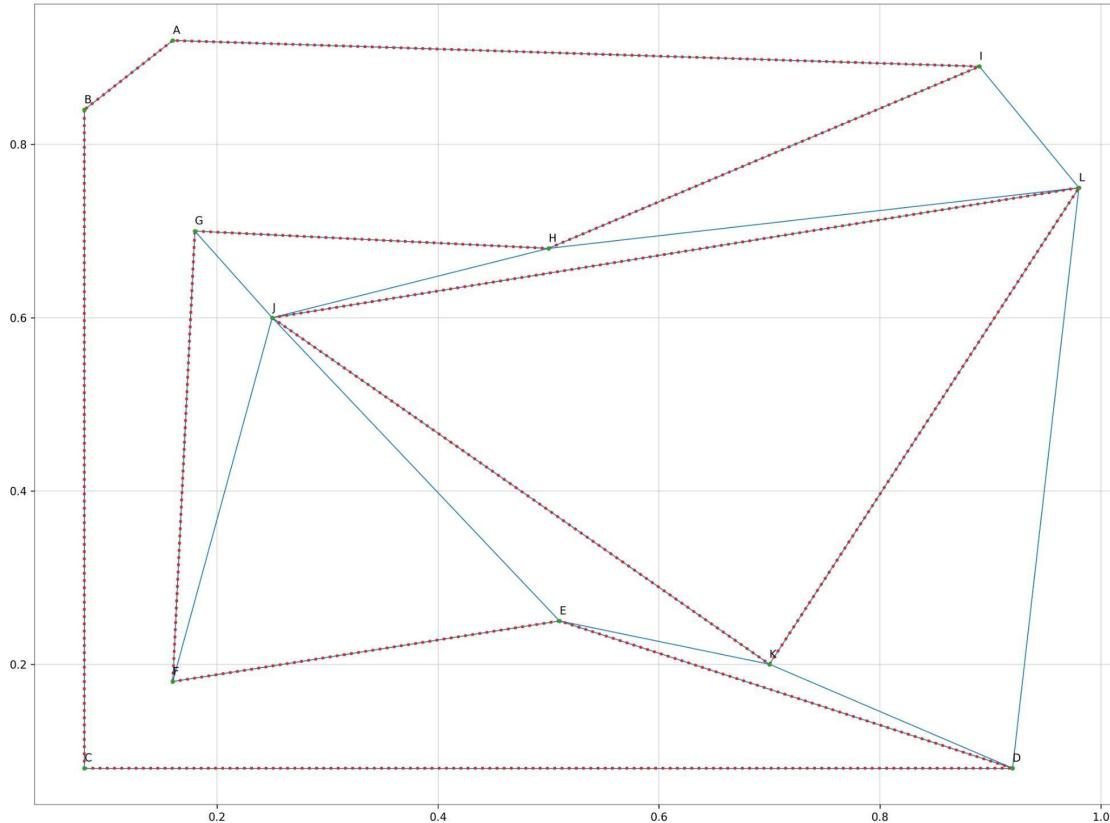
Wierzchołek **E** widoczny, gdy miotła pokrywa się z krawędzią **BC**.



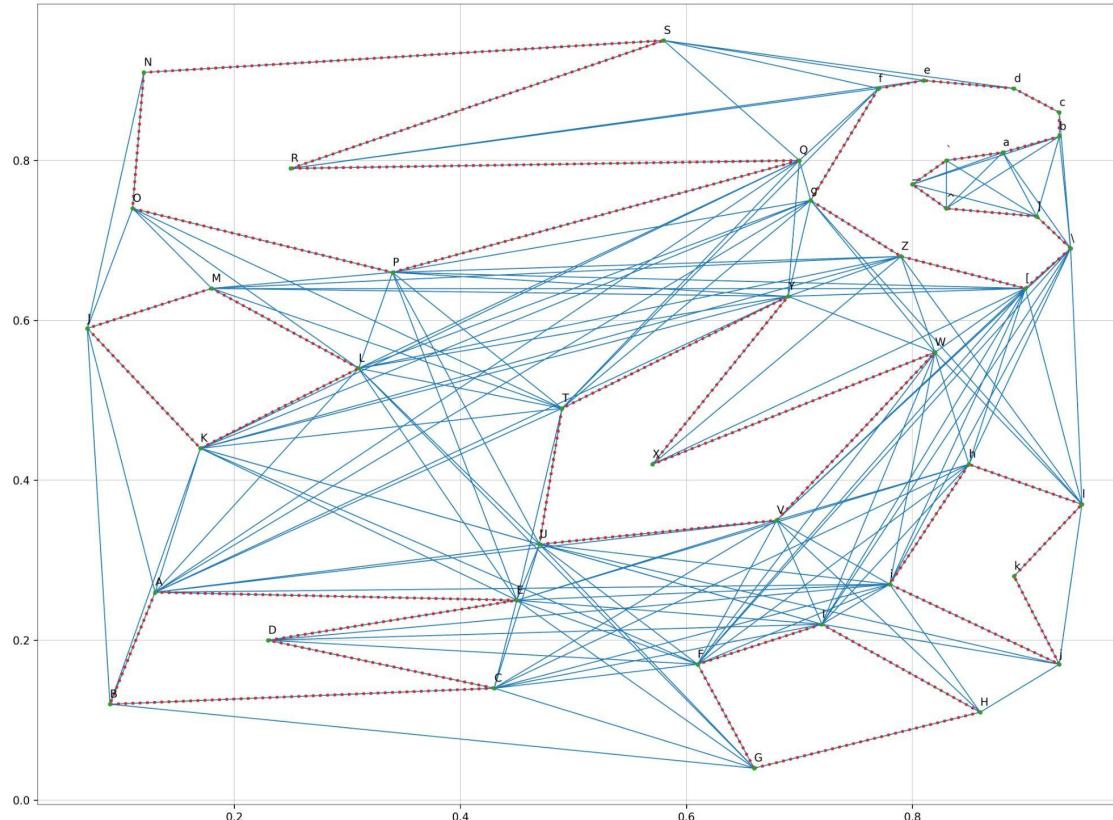
Przykłady działania - graf widoczności



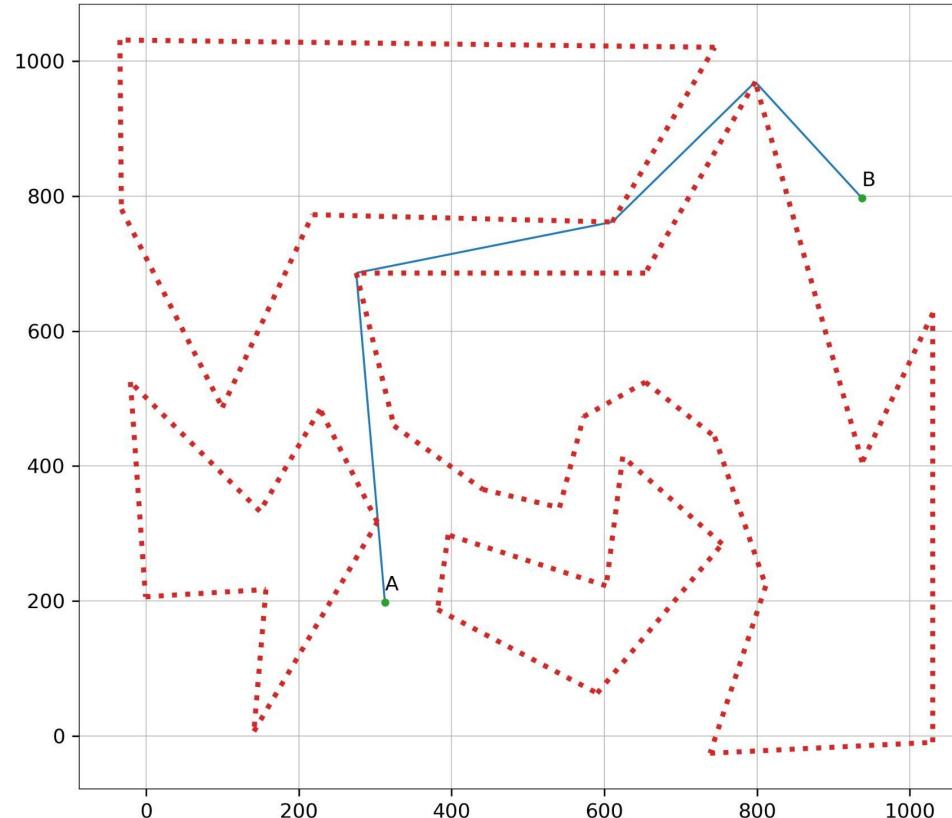
Przykłady działania - graf widoczności



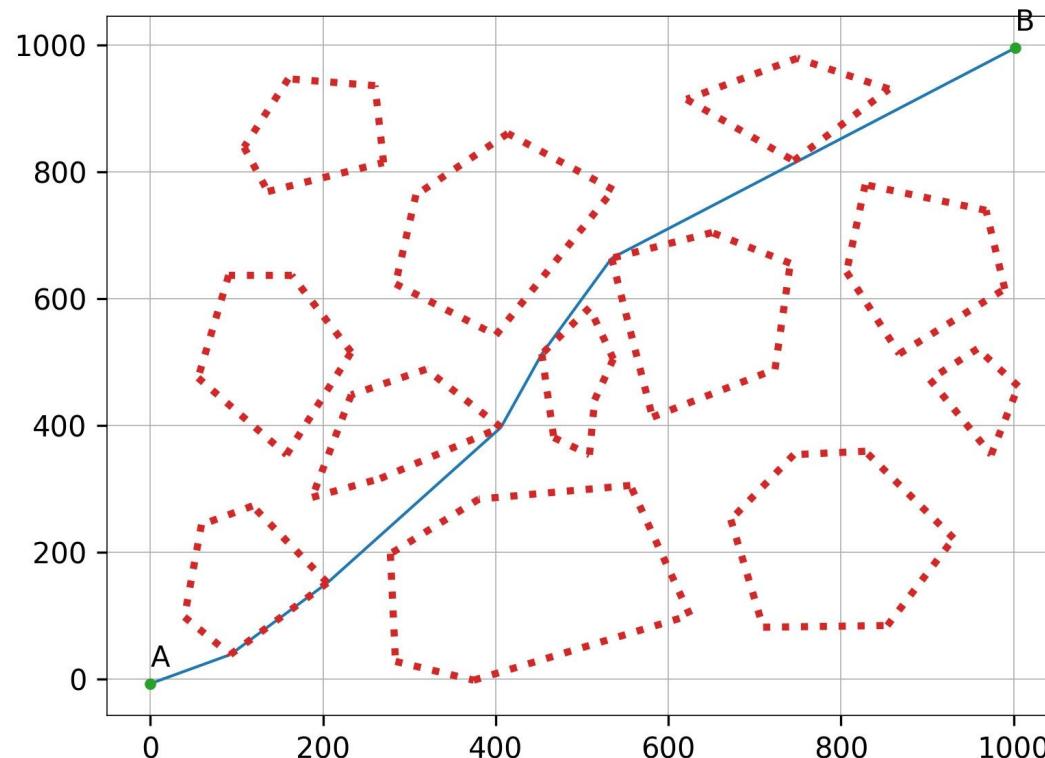
Przykłady działania - graf widoczności



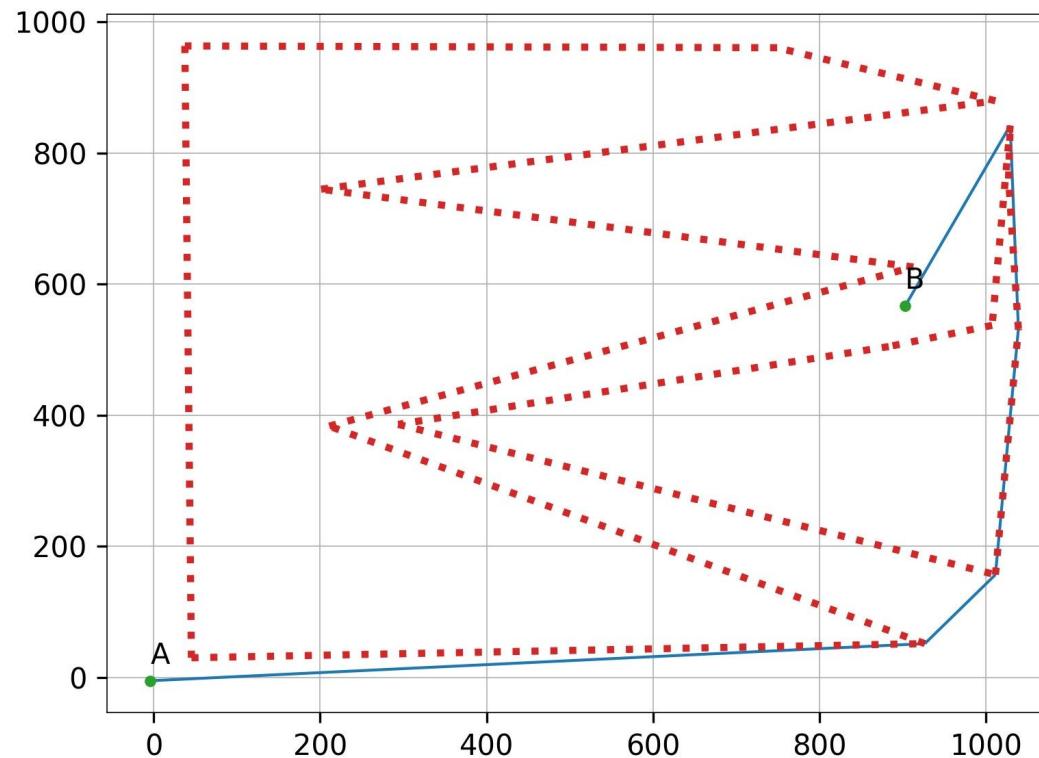
Przykłady działania - najkrótsza ścieżka



Przykłady działania - najkrótsza ścieżka



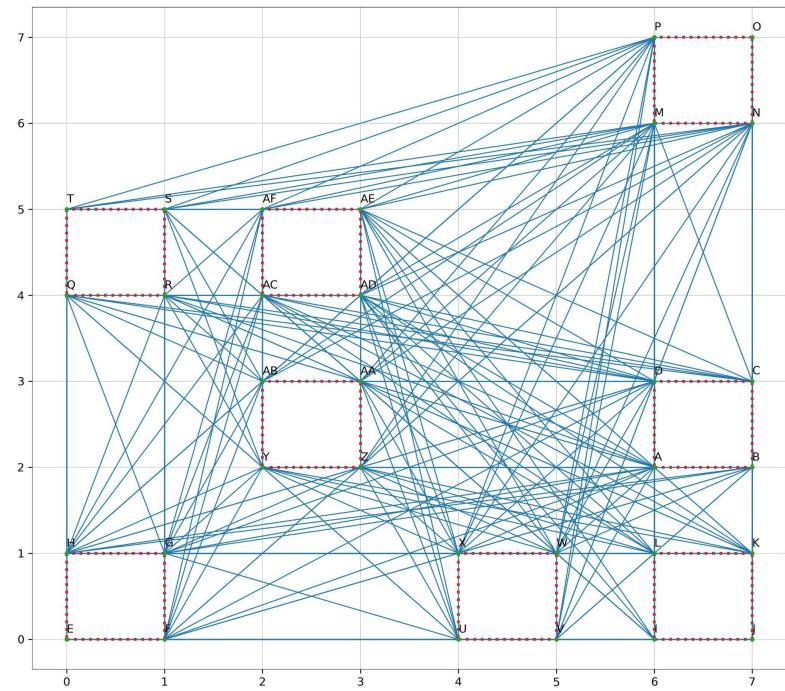
Przykłady działania - najkrótsza ścieżka



Przykłady działania - losowo generowane kwadraty

Przykład działania dla losowej siatki 8 kwadratów na planszy 7x7.

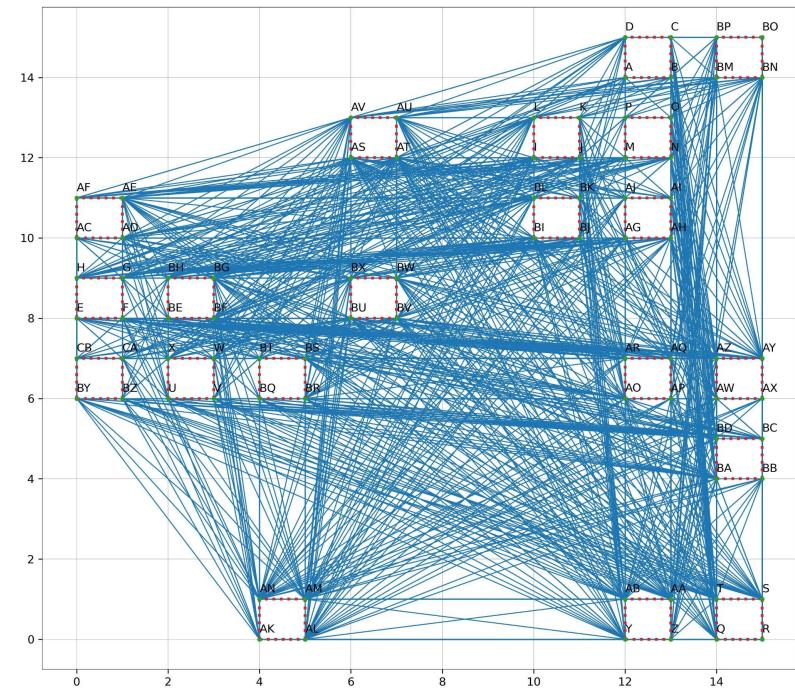
Czas wykonania: **0.04488** sekund



Przykłady działania - losowo generowane kwadraty

Przykład działania dla losowej siatki 20 kwadratów na planszy 16x16

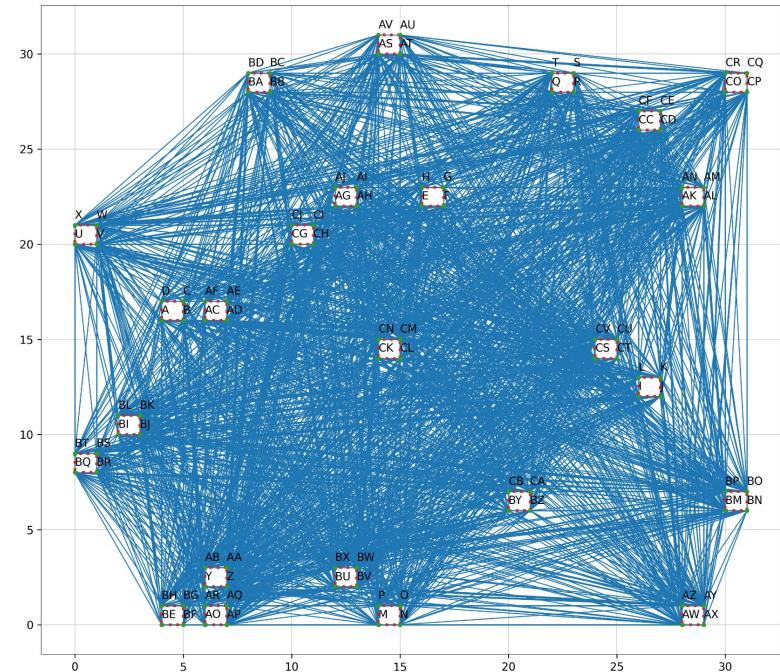
Czas wykonania: **0.31419** sekund



Przykłady działania - losowo generowane kwadraty

Przykład działania dla losowej siatki 25 kwadratów na planszy 31x31

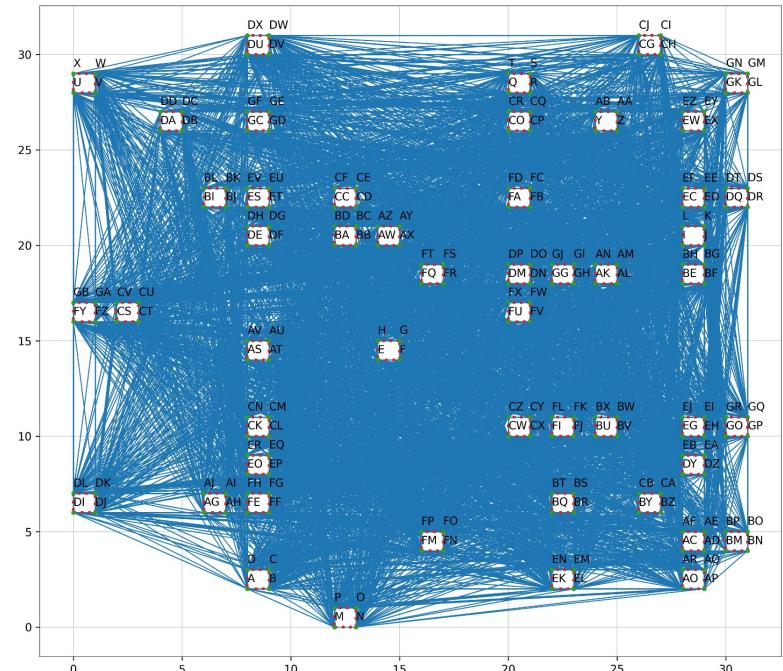
Czas wykonania: **0.48368** sekund



Przykłady działania - losowo generowane kwadraty

Przykład działania dla losowej siatki 50 kwadratów na planszy 31x31

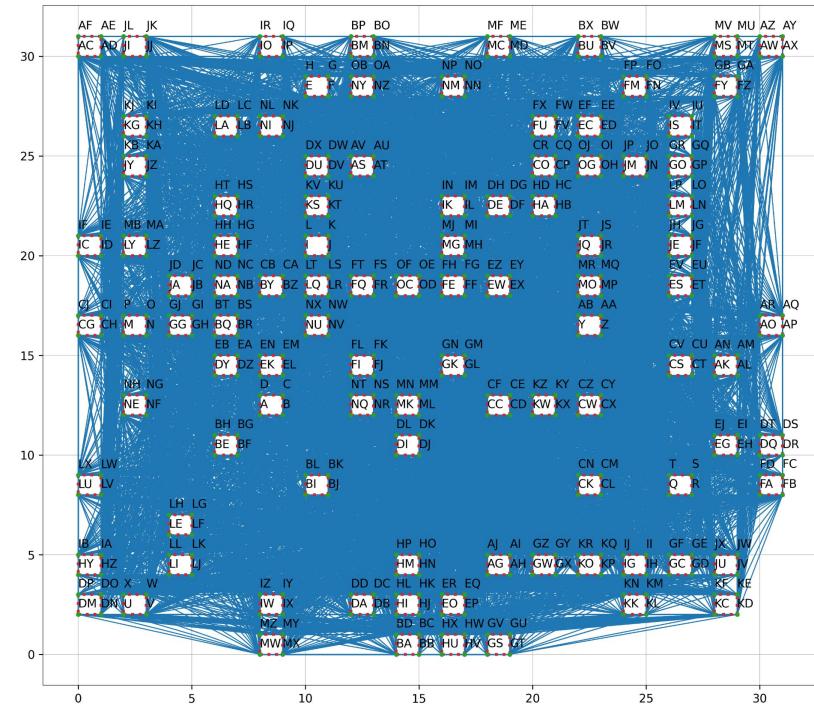
Czas wykonania: **2.2919** sekund



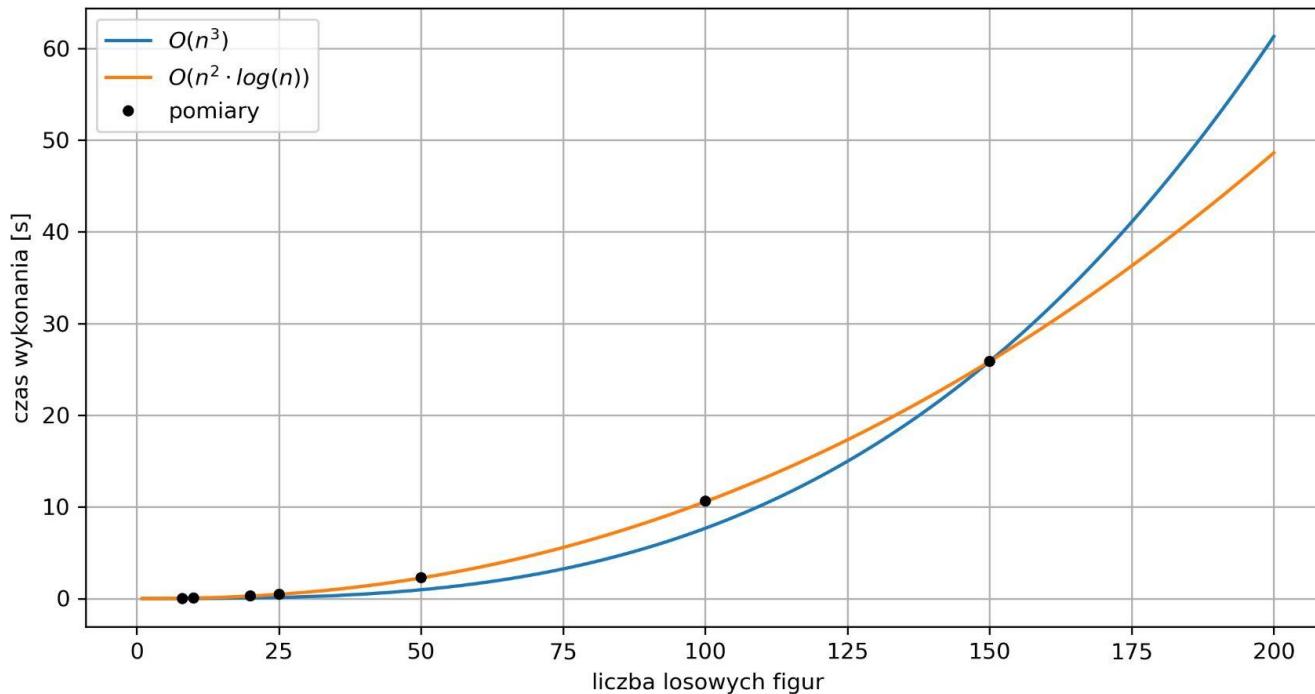
Przykłady działania - losowo generowane kwadraty

Przykład działania dla losowej siatki 100 kwadratów na planszy 31x31

Czas wykonania: **10.62454** sekund



Wykres czasów wykonania dla testów losowych



Bibliografia

Algorytm został przygotowany na podstawie książki Marka de Berga pt. *Geometria Obliczeniowa - Algorytmy i Zastosowania.*