1. BFS dla macierzowej reprezentacji grafu skierowanego.

Panstwa implementacja powinna zwracac tablice postaci

[(*parent*0*, d*0)*,* (*parent*1*, d*1)*, . . . ,* (*parentn−*1*, dn−*1)]*,*

gdzie *parenti* to poprzednik wierzchołka *i* na najkrótszej sciezce z wierzchołka

zródłowego a *di* to odległosc *i*-go wierzchołka od zródłowe.

1. Prosze zaimplementowac algorytm DFS dla grafu nieskierowanego reprezentowanego

przez listy sasiedztwa. Panstwa implementacja powinna zwracac tablice

postaci [*parent*0*, parent*1*, . . . , parentn−*1], gdzie *parenti* to poprzednik wierzchołka

*i* w drzewie DFS.

1. Prosze zaimplementowac algorytm, który majec na wejsciu napis

zbudowany z nawiasow (, ), [, i ] sprawdza czy nawiasowanie jest poprawne

.

1. Jak zaimplementowac kolejke na dwóch stosach?
2. Prosze podac algorytm “przesuwajacy” zadana *n*-elementowa tablice

*A* o *k* pozycji. (Przesuniecie tablicy oznacza, ze element, który był pierwotnie na pozycji *i*, powinien sie

znalezc na pozycje *n* + *k* (modulo *n*). Algorytm powinien działac w miejscu.

1. **(kapitan statku) (implementacja)** Kapitan pewnego statku

zastanawia sie, czy moze wpłynac do portu mimo, ze nastapił odpływ. Do dyspozycji ma mape zatoki w

postaci tablicy *M*, gdzie *M*[*y*][*x*] to głebokosc zatoki na pozycji (*x, y*). Jesli jest ona wieksza niz pewna

wartosc int T to statek moze sie tam znalezc. Poczatkowo statek jest na pozycji (0*,* 0) a port znajduje sie na

pozycji (*n−*1*,m−*1). Z danej pozycji statek moze przepłynac bezposrednio jedynie na pozycje bezposrednio

obok (to znaczy, na pozycje, której dokładnie jedna ze współrzednych rózni sie o jeden). Prosze napisac

funkcje rozwiazujaca problem kapitana.

1. Prosze zaimplementowac nastepujace algorytmy: SPRAWDZIC

1. Sprawdzanie czy graf nieskierowany jest dwudzielny (czyli czy da sie podzielic jego wierzchołki na dwa

zbiory, takie ze krawedzie łacza jedynie wierzchołki z róznych zbiorów).

2. Sprawdzanie czy graf nieskierowany posiada cykl.

1. **(uniwersalne ujscie)** Mówimy, ze wierzchołek *t* w grafie skierowanym jest uniwersalnym

ujsciem, jesli (a) z kazdego innego wierzchołka *v* istnieje krawedz z *v* do *t*, oraz (b) nie istnieje zadna krawedz

wychodzaca z *t*. Prosze podac algorytm znajdujacy ujscie (jesli istnieje) przy reprezentacji macierzowej grafu.

1. Dany jest ciag przedziałów postaci [*ai, bi*]. Dwa przedziały mozna

skleic jesli maja dokładnie jeden punkt wspólny. Prosze wskazac algorytmy dla nastepujacych problemów:

1. Problem stwierdzenia, czy da sie uzyskac przedział [*a, b*] przez sklejanie odcinków.

2. Problem stwierdzenia jaki najdłuzszy odcinek mozna uzyskac sklejajac najwyzej *k* odcinków.

1. **(czy nieskierowany?)** Prosze podac algorytm, który majac na wejsciu graf *G* reprezentowany

przez listy sasiedztwa sprawdza, czy dla kazdej krawedzie *u ! v* istnieje takze krawedz przeciwna.

1. **(malejace krawedzie) (implementacja)** Dany jest graf *G* = (*V,E*), gdzie kazda krawedz

ma wage ze zbioru *{*1*, . . . , |E|}* (wagi krawedzi sa parami rózne). Prosze zaproponowac algorytm, który dla

danych wierzchołków *x* i *y* sprawdza, czy istnieje sciezka z *x* do *y*, w której przechodzimy po krawedziach o

coraz mniejszych wagach.

1. **(Pause)** Znany operator telefonii komórkowej Pause postanowił zakonczyc działalnosc w

Polsce. Jednym z głównych elementów całej procedury jest wyłaczenie wszystkich stacji nadawczych (które

tworza spójny graf połaczen). Ze wzgledów technologicznych urzadzenia nalezy wyłaczac pojedynczo a operatorowi dodatkowo zalezy na tym, by podczas całego procesu wszyscy abonenci znajdujacy sie w zasiegu

działajacych stacji mogli sie ze soba łaczyc(czyli by graf pozostał spójny). Prosze zaproponowac algorytm

podajacy kolejnosc wyłaczania stacji.

1. Dana jest szachownica o wymiarach *n×n*. Kazde pole (*i, j*) ma koszt (liczbe ze zbioru *{*1*, . . . ,* 5*}*) umieszczony

w tablicy *A* (na polu *A*[*j*][*i*]).Wlewym górnym rogu szachownicy (na pozycji (0*,* 0) stoi król, którego zadaniem

jest przejasc do prawego dolnego rogu, przechodzac po polach o minimalnym sumarycznym koszcie (jesli król

stoi na polu (i,j) to ponosi koszt *A*[*j*][*i*]; tak wiec kazda trasa zawiera koszt *A*[0][0] i *A*[*n −* 1][*n −* 1]). Prosze

zaimplementowac funkcje kings path(A), która oblicza koszt sciezki króla. Funkcja powinna byc mozliwie

jak najszybsza (w szczególnosci oczekujemy złozonosci *O*(*n*2)).

Panstwa kod powinien miec nastepujaca postac (bedzie uruchamiany; prosze nie usuwac fragmentu testujacego;

sprawdzajacy moze takze dołozyc swoje testy):

def kings\_path( A ):

# policz koszt trasy

A = [[1,1,2],

[5,1,3],

[4,1,1]]

print( kings\_path( A ) ) # wypisze 5

1. Prosze zaimplementowac funkcje knapsack2d(P, max w, max h), która oblicza maksymalna wartosc plecaka

w dwuwymiarowej wersji dyskretnego problemu plecakowego, okreslonego nastepujaco. Mamy dana tablice

*n* trójek *P* = [(*v*0*,w*0*, h*0)*, . . .* (*vn−*1*,wn−*1*, hn−*1)], gdzie *i*-ta krotka ma nastepujace znaczenie: *vi* to wartosc

*i*-go przemiotu, *wi* to jego waga, a *hi* to jego wysokosc. Złodziej chce wybrac przedmioty o maksymalnej

symarycznej wartosci, których łaczna waga nie przekracza danej liczby max w oraz których łaczna wysokosc

nie przekracza danej liczby max h (przedmioty zapakowane sa w kartony, które złodziej układa jeden na

drugim).

Panstwa kod powinien miec nastepujaca postac (bedzie uruchamiany; prosze nie usuwac fragmentu testujacego;

sprawdzajacy moze takze dołozyc swoje testy):

def knapsack2d( V, max\_w, max\_h ):

# tu prosze umiescic swoja implementacje

P = [(5,10,3), (7,8,12), (2,7,3)]

print( knapsack2d( P, 16, 15 ) # wypisze 9

**Cwiczenia 8: Algorytmy dynamiczne, zachłanne i podstawy grafowych**

1. **(kolorowanie grafu)** Mamy dany nieskierowany graf *G* = (*V,E*).

Dla kazdego wierzchołka *v* nalezy wybrac liczbe naturalna *f*(*v*) (nazywana kolorem *v*) tak, zeby dla kazdej

krawedzi *{x, y} 2 E* zachodzilo, ze *f*(*x*) *6*= *f*(*y*). Nalezy uzyc jak najmniej kolorów (czyli zbior *f*(*V* ) powinien

miec minimalna licznosc). Problem jest NP-zupełny wiec nie istnieje optymalny algorytm wielomianowy (o

ile P jest rózne od NP). Prosze podac algorytm zachłanny, który uzywa najwyzej *D*+1 kolorów, gdzie *D* to

maksymalny stopien wierzchołka w *G*.

1. Dana jest tablica *A*[*n*] z długosciami samochodów, które stoja w kolejce,

zeby wjechac na prom. Prom ma dwa pasy (lewy i prawy), oba długosci *D*. Prosze napisac program, który

wyznacza, które samochody powinny pojechac na który pas, zeby na promie zmiesciło sie jak najwiecej aut.

Auta musza wjezdzac w takiej kolejnosci, w jakiej sa podane w tablicy *A*.

1. Dany jest zbiór przedziałów *I* = *{*[*a*1*, b*1]*, . . . ,* [*an, bn*]*}*. Mówimy, ze przedział

[*ai, bi*] deaktywuje przedział [*aj , bj* ] jesli *ai ­ aj* oraz *bi ¬ bj* (czyli pierwszy jest podzbiorem drugiego). Prosze

zaproponowac algorytm, który znajduje podzbiór *I* o maksymalnym rozmiarze taki, ze zaden przedział nie

jest deaktywowany przez inny (innymi słowy, nalezy dany przedział usunac jesli jest nadzbiorem innego, a

sposród identycznych przedziałów usunac wszystkie poza jednym).

1. Dany jest zbiór zadan *T* = *{t*1*, . . . , tn}*. Kazde zadanie *ti* ma

podany czas rozpoczecia *si 2* R oraz czas zakonczenia *ei 2* R. Prosze zaproponowac algorytm (bez implementacji),

który znajduje taki podzbiór *k* zadan (gdzie *k 2* N to dany parametr wejsciowy), ze (a) zadne

dwa zadania na siebie nie nachodza oraz (b) czas jaki mija od rozpoczecia najwczesniejszego zadania do

zakonczenia najpózniejszego jest minimalny. Jesli podzbioru rozmiaru *k* spełniajacego warunki zadania nie

ma, to algorytm powinien to stwierdzic. Algorytm powinien byc jak najszybszy. Mozna załozyc, ze zaden

przedział nie jest podzbiorem innego.

1. Dana jest mapa kraju w postaci grafu *G* = (*V,E*). Kierowca chce przejechac

z miasta (wierzchołka) *s* to miasta *t*. Niestety niektóre drogi (krawedzie) sa płatne. Kazda droga ma taka

sama jednostkowa opłate. Prosze podac algorytm, który znajduje trase wymagajaca jak najmniejszej liczby

opłat. W ogólnosci graf *G* jest skierowany, ale mozna najpierw wskazac algorytm dla grafu nieskierowanego.

1. Dany jest graf *G* = (*V,E*), którego wierzchołki reprezentuja punkty

nawigacyjne nad Bajtocja, a krawedzie reprezentuja korytarze powietrzne miedzy tymi punktami. Kazdy

korytarz powietrzny *ei 2 E* powiazany jest z optymalnym pułapem przelotu *pi 2* N (wyrazonym w metrach).

Przepisy dopuszczaja przelot danym korytarzem jesli pułap samolotu rózni sie od optymalnego najwyzej o *t*

metrów. Prosze zaproponowac algorytm (bez implementacji), który sprawdza czy istnieje mozliwosc przelotu

z zadanego punktu *x 2 V* do zadanego punktu *y 2 V* w taki sposób, zeby samolot nigdy nie zmieniał pułapu.

Algorytm powinien byc poprawny i mozliwie jak najszybszy. Prosze oszacowac jego złozonosc czasowa.

1. **(bitoniczny problem komiwojazera)** Mamy dane punkty na płaszczyznie. Komiwojazer

moze sie poruszac najpierw tylko w prawo a potem tylko w lewo (oraz dowolnie góra/dół). Nalezy podac

algorytm dynamiczny znajdujacy optymalna trase komiwojazera, w której odwiedza wszystkie zadane punkty

i wraca do punktu startu (czyli do punktu maksymalnie na lewo).

1. **(problem komiwojazera)** Na wejsciu dostajemy zbiór *n* miast oraz macierz odległosci

miedzy kazda para miast (powiedzmy, ze *dij* to odległosc z miasta *i* do *j*, oraz *dij* = *dji*). Zadanie polega na

obliczeniu najkrótszej trasy, która zaczyna sie w miescie 1, konczy w miescie 1 i przebiega przez wszystkie

pozostałe, zadnego nie powtarzajac.