1. Prosze zaimplementowac rozwiazanie problemu plecakowego przy pomocy rekurencji

ze spamietywaniem.

1. Prosze zaimplementowac rozwiazanie problemu plecakowego tak, zeby funkcja

zwracały liste indeksów przedmiotów, które nalezy wybrac (mozna korzystac z

funkcji append do dopisywania elementów na koncu listy).

1. Prosze zaimplementowac rozwiazanie problemu “Impraza firmowa” tak, by zwracane

były imiona pracowników, którzy ida na impreze. Nalezy załozyc, ze pracownicy

reprezentowani sa w strukturze:

class Employee:

def \_\_init\_\_(self, fun, name):

self.emp = []

self.fun = fun

self.name = name

# wolno dokładac własne pola do struktury

1. **(problem sumy podzbioru)** Dana jest tablica *n* liczb *A*. Prosze podac i zaimplementowac

algorytm, który sprawdza, czy da sie wybrac podciag liczb z *A*, które sumuja sie do zadanej wartosci *T*.

1. **(najdłuzszy wspólny podciag)** Mamy dane dwie tablice, *A*[*n*] i *B*[*n*]. Nalezy znalezc długosc

ich najdłuzszego wspólnego podciagu. (Klasyczny algorytm dynamiczny *O*(*n*2)).

1. **(najdłuzszy podciag rosnacy)** Prosze rozwiazac dwa nastepujace zadania:

Jak wykorzystac algorytm dla problemu najdłuzszego wspólnego podciagu do rozwiazania zadania

najdłuzszego rosnacego podciagu?

Na wykładzie podalismy algorytm działajacy w czasie *O*(*n*2). Prosze podac algorytm o złozonosci

*O*(*n* log *n*).

1. **(mnozenie macierzy)** Dany jest cieg macierzy *A*1*,A*2*, . . . ,An*. Ktos chce policzyc iloczyn

*A*1*A*2 *· · ·An*. Macierze nie sa koniecznie kwadratowe (ale oczywiscie znamy ich rozmiary). Zaleznie w jakiej

kolejnosci wykonujemy mnozenia, koszt obliczeniowy moze byc rózny—nalezy podac algorytm znajdujacy

koszt mnozenia przy optymalnym doborze kolejnosci.

1. **(wedrówka po szachownicy)** Dana jest szachownica *A* o wymiarach *n × n*. Szachownica

zawiera liczby wymierne. Nalezy przejsc z pola (1*,* 1) na pole (*n, n*) korzystajac jedynie z ruchów “w dół”

oraz “w prawo”. Wejscie na dane pole kosztuje tyle, co znajdujaca sie tam liczba. Prosze podac algorytm

znajdujacy trase o minimalnym koszcie.

1. **(wydawanie monet)** Mamy dana tablice z nominałami monet stosowanych w pewnym dziwnym

kraju, oraz kwote *T*. Prosze podac algorytm, który oblicza minimalna ilosc monet potrzebna do wydania

kwoty *T* (algorytm zachłanny, wydajacy najpierw najwieksza monete, nie działa: dla monet 1, 5, 8 wyda

kwote 15 jako 8 + 5 + 1 + 1 zamiast 5 + 5 + 5).

1. Dana jest *n* elementowa tablica *A* = [(*b*1*, e*1)*, . . . ,* (*bn, en*)], gdzie kazda para

(*bi, ei*) oznacza zajecia rozpoczynajace sie w chwili *bi* i konczace w chwili *ei*.

Prosze zaimplementowac funkcje tasks(A), która zwraca ile maksymalnie zajec

mozna wybrac tak, by na siebie nie nachodziły. Mozna załozyc, ze wszystkie

liczby w tablicy *A* sa naturalne. Przedziały nalezy traktowac jako otwarte, czyli

np. zajecia (1*,* 3) oraz (3*,* 5) nie nachodza na siebie.

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

usuwac fragmentu testujacego; sprawdzajacy moze takze dołozyc swoje testy):

def tasks(A):

# tu prosze umiescic swoja implementacje

# elementarny test, powinien wypisac 2

print( tasks([ (0,10), (10,20), (5,15) ] )

1. **(kody Huffmana)**

Dana jest tablica *n* liczb naturalnych *A*. Liczba *A*[*i*] mówi ile razy *i*-ty symbol

pojawia sie w tekscie. Prosze zaimplementowac funkcje huffman len(A), która

oblicza ile bitów zajełoby zapisanie tekstu składajacego sie własnie z takiej liczby

symboli, jesli uzytoby optymalnego kodu Huffmana. Funkcja powinna działac

w czasie *O*(*n* log *n*). Podpowiedz: Moze sie przydac struktura kopca.

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

usuwac fragmentu testujacego; sprawdzajacy moze takze dołozyc swoje testy):

def huffman\_len(A):

# tu prosze umiescic swoja implementacje

# elementarny test, powinien wypisac 2600

print( huffman\_len([ 200, 700, 180, 120, 70, 30] )

1. **(ciagły problem plecakowy)**

Dana jest *n* elementowa tablica *A* = [(*P*1*,W*1)*, . . . ,* (*Pn,Wn*)] opisujaca egzemplarz

ciagłego problemu plecakowego; *A* opisuje dostepne płyny a *k* objetosc

plecaka (a raczej pojemnika; *k* jest podane w litrach). Dla *i*-go przedmiotu *Pi*

oznacza jego wartosc za wszystkie dostepne *Wi* litrów. Prosze zaimplementowac

funkcje knapsack(A,k), która oblicza wartosc najlepszego pojemnika, jaki

mozna uzyskac.

1. **(problem stacji benzynowych)** Pewien podróznik chce przebyc trase z punktu *A* do punktu

*B*. Niestety jego samochód spala dokładnie jeden litr paliwa na jeden kilometr trasy (mozna powiedziec, ze

jedzie czołgiem... znaczenie punktów *A* i *B* w ramach obecnej sytuacji geopolitycznej wybierzcie sobie sami).

W baku miesci sie dokładnie *D* litrów paliwa. Trasa z *A* do *B* to prosta, na której znajduja sie stacje

benzynowe. Mamy dwa rózne zadania (rozwiazywane osobno):

(1) wyznaczyc trase, na której tankujemy minimalna liczbe razy.

(2) wyznaczyc trase, której koszt jest minimalny (wówczas znamy jeszcze dla kazdej stacji cene za litr

paliwa, nie musimy zawsze tankowac do pełna).

(3) Bonus: j.w., ale jesli na stacji tankujemy, to musimy zatankowac do pełna.

1. **(wybór zadan z terminami)** Mamy dany zbiór zadan *T* = *{t*1*, . . . , tn}*. Kazde zadanie *ti*

dodatkowo posiada: (a) termin wykonania *d*(*ti*) (liczba naturalna) oraz (b) zysk *g*(*ti*) za wykonanie w terminie

(liczba naturalna). Wykonanie kazdego zadania trwa jednostke czasu. Jesli zadanie *ti* zostanie wykonane

przed przekroczeniem swojego terminu *d*(*ti*), to dostajemy za nie nagrode *g*(*ti*) (pierwsze wybrane zadanie

jest wykonywane w chwili 0, drugie wybrane zadanie w chwili 1, trzecie w chwili 2, itd.).

Prosze podac algorytm, który znajduje podzbiór zadan, które mozna wykonac w terminie i który prowadzi

do maksymalnego zysku. Prosze uzasadnic poprawnosc algorytmu.

1. **Zadanie 3. (ładowanie przyczepy)** Mamy przyczepe o pojemnosci *K* kilogramów oraz zbiór ładunków o

wagach *w*1*, . . . ,wn*. Waga kazdego z ładunków jest potega dwójki (czyli, na przykład, dla siedmiu ładunków

wagi moga wynosic 2, 2, 4, 8, 1, 8, 16, a pojemnosc przyczepy *K* = 27). Prosze podac algorytm zachłanny (i

uzasadnic jego poprawnosc), który wybiera ładunki tak, ze przyczepa jest mozliwie maksymalnie zapełniona

(ale bez przekraczania pojemnosci) i jednoczesnie uzylismy mozliwie jak najmniej ładunków. (Ale jesli da sie

np. załadowac przyczepe do pełna uzywajac 100 ładunków, albo zaladowac do pojemnosci *K −* 1 uzywajac

jednego ładunku, to lepsze jest to pierwsze rozwiazanie).

1. **Zadanie 4. (pokrycie przedziałami jednostkowymi)** Dany jest zbiór punktów *X* = *{x*1*, . . . , xn}* na

prostej. Prosze podac algorytm, który znajduje minimalna liczbe przedziałów jednostkowych domknietych,

potrzebnych do pokrycia wszystkich punktów z *X*. (Przykład: Jesli *X* = *{*0*.*25*,* 0*.*5*,* 1*.*6*}* to potrzeba dwóch

przedziałów, np. [0*.*2*,* 1*.*2] oraz [1*.*4*,* 2*.*4]).

1. **Zadanie 5. (wieze)** Grupa *m* dzieci bawi sie w układanie mozliwie jak najwiekszej wiezy. Kazde dziecko

ma klocki róznej wysokosci. Pierwsze dziecko ma klocki o wysokosciach *w*1

1*, . . . ,w*1*n*

1 , drugie dziecko klocki o

wyskosciach *w*2

1*, . . . ,w*2*n*

2 , itd. Dzieci własnie poszły zjesc deser zanim ułoza swoje wieze, ale jedno dziecko

zostało. Ma teraz jedyna okazje, zeby podebrac kilka klocków innym dzieciom tak, zeby jego wieza była

najwyzsza. Prosze podac mozliwie najszybszy algorytm rozwiazujacy ten problem, który zabiera minimalna

ilosc klocków. (Prosze zwrócic uwage, ze liczby *wij*

moga byc bardzo duze.)

1. **(suma odległosci)** Dana jest posortowana tablica A zawierajaca *n* liczb i celem jest wyznaczenie

liczby *x* takiej, ze wartosc

P*n−*1

*i*=0 *|A*[*i*]*−x|* jest minimalna. Prosze zaproponowac algorytm, uzasadnic

jego poprawnosc oraz ocenic złozonosc obliczeniowa.

1. W jednej z chińskich prowincji postanowiono wybudować serię maszyn chroniących ludność przed koronawirusem. Prowincję można zobrazować jako tablicę wartości 1 i 0, gdzie arr[i] = 1 oznacza, że w mieście i można zbudować maszynę, a wartość 0, że nie można. Dana jest również liczba k, która oznacza, że jeśli postawimy maszynę w mieście i, to miasta o indeksach j takich, że abs(i-j) < k są przez nią chronione. Należy zaproponować algorytm, który stwierdzi ile minimalnie maszyn potrzeba aby zapewnić ochronę w każdym mieście, lub -1 jeśli jest to niemożliwe.
2. Dany jest zbiór przedziałów otwartych. Zaproponuj algorytm, który znajdzie podzbiór tego zbioru, taki że:

jego rozmiar wynosi dokładnie k

przedziały są rozłączne

różnica między najwcześniejszym początkiem, a najdalszym końcem jest minimalna.

Jeśli rozwiązanie nie istnieje, to algorytm powinien to stwierdzić. Algorytm powinien być w miarę możliwości szybki, ale przede wszystkim poprawny.

1. Dany jest string, w którym niektóre litery się powtarzają. Należy zaproponować algorytm, który usunie ze stringa duplikaty tak, że otrzymany string będzie leksykograficznie najmniejszy.

Przykład: cbacdcbc, odpowiedzią jest acbd.

1. Dana jest lista zleceń. Każde zlecenie wymaga pewnego kapitału początkowego Ci, który należy mieć, żeby zacząć zlecenie oraz zysk Pi, który doda się do naszego całkowitego kapitału, gdy wykonamy zlecenie. Mając kapitał początkowy W i liczbę k wybierz co najwyżej k zleceń tak, że skończysz z maksymalnym możliwym kapitałem.

Przykład: k = 2, W = 0, P=[1,2,3], C=[0,1,1]. Rozwiązanie: na początku mamy kapitał 0, więc możemy wybrać tylko zlecenie pierwsze. Po jego ukończeniu mamy kapitał równy 1, więc możemy wybrać albo zlecenie 2 albo 3. Zlecenie 3 ma większy profit więc wybieramy zlecenie 3, ponieważ możemy wybrać już tylko 1 zlecenie (k = 2). Kończymy z kapitałem 4.