410. Udowodnić, że jeśli $f : [1, \infty) \to \mathbb{R}$ jest funkcją niemalejącą, wklęsłą, klasy C^2 to ciąg

$$a_n = \int_1^n f(x)\,dx - \sum_{k=1}^{n} f(k) + \frac{1}{2} f(n)$$

jest ograniczony.

411. Wyznacz wartość całki

$$\int_1^2 \left(e^{1-\frac{1}{(x-1)^2}} + 1\right) + \left(1 + \frac{1}{\sqrt{1 - \log(x-1)}}\right) dx.$$

412. Wykorzystując oszacowania wykorzystane w dowodzie wzoru Stirlinga na wykładzie wyznaczyć liczbę cyfr liczb 1000!.

413. Niech $f : [0, \infty) \to \mathbb{R}$ będzie funkcją ciągłą o wartościach dodatnich. Wykazać, że dla każdego $x > 0$ prawdziwa jest nierówność

$$\int_0^x t^2 f(t)\,dt \cdot \int_0^x f(t)\,dt \geq \left(\int_0^x t f(t)\,dt\right)^2.$$

414. Załóżmy, że $f : [0, 2\pi]$ spełnia warunek Lipshitza. Wykazać, że istnieje $C > 0$, taka że dla $k \in \mathbb{N}$ spełnione jest szacowanie[4]:

$$\left|\int_0^{2\pi} f(x) \sin(kx)\,dx\right| \leq \frac{C}{k}.$$

[4] *Wskazówka:* Może pomóc zrobienie najpierw tego zadania dla funkcji klasy C^1 przy pomocy całkowania przez części. Jeśli nie założymy różniczkowalności można rozważać sumy Riemmana i zastosować odpowiednik całkowania przez części dla sum, czyli sumowanie Abela.

Lista zadań. Nr 2. 13 marca 2025

Algorytmy i Struktury Danych

IIUWr.

0. (0pkt) Przeczytaj notatkę do wykładu o algorytmach zachłannych.
1. (1pkt) Przeprowadź dowód poprawności algorytmu Kruskala, który przyrównuje ciąg krawędzi drzewa wybranych przez algorytm Kruskala z ciągiem krawędzi minimalnego drzewa spinającego (otrzymanego przez jakiś algorytm optymalny). W dowodzie nie powołuj się na własności typu *cut property* czy *cycle property*.
2. (1pkt) Danych jest n odcinków $I_j = \langle p_j, k_j \rangle$, leżących na osi OX, $j = 1, \ldots, n$. Ułóż algorytm znajdujący zbiór $S \subseteq \{I_1, \ldots, I_n\}$, nieprzecinających się odcinków, o największej mocy.
3. (1,5pkt) Rozważ wersję problemu wydawania reszty, w którym i-ty nominał ma wartość równą i-tej liczbie Fibonacciego ($i = 1, 2, \ldots$). Ułóż algorytm, który, wydając resztę, używa co najwyżej jednej monety każdego nominału. Czy taki algorytm jest w stanie wydać każdą kwotę? Czy liczba monet wydana przez Twój algorytm jest zawsze optymalna (tj. minimalna)?
4. (2pkt) Masz początkowo do dyspozycji m monet o nominale 1 i nieskończoną liczbę monet o nominale 100. W kolejnych n dniach masz zrobić zakupy w ulubionym sklepie (w i-tym dniu zakup o wartości c_i). Jeśli w i-tym dniu nie odliczysz dokładnej kwoty (tj. c_i), kasa wyda Ci dokładną wartość reszty, używając minimalnej ilości monet (kasa także używa jedynie monet o nominałach 1 i 100), a Twoja "atrakcyjność kliencka" zostanie zmniejszona o wartość $x \cdot w_i$, gdzie x jest liczbą monet wydanych przez kasę a w_i jest współczynnikiem używanych przez sklep w i-tym dniu.

   Ułóż algorytm obliczający o ile co najmniej zmniejszy się Twoja atrakcyjność kliencka po dokonaniu wszystkich zakupów. Możesz przyjąć, że c_i oraz w_i są liczbami naturalnymi nie większymi od n.
5. (1,5pkt) Ułóż algorytm, który dla danego grafu $G = V, E)$ oraz liczby naturalnej k znajdzie możliwie największy podzbiór $V' \subseteq V$, taki, że dla każdego wierzchołka $v \in V'$ zachodzi:

   $$|\{u \in V' : \{v, u\} \in E\}| \geq k \text{ oraz } |\{u \in V' : \{v, u\} \notin E\}| \geq k.$$
6. (1,5pkt) Ułóż algorytm, który dla danego n-wierzchołkowego drzewa i liczby k, pokoloruje jak najwięcej wierzchołków tak, by na każdej ścieżce prostej było nie więcej niż k pokolorowanych wierzchołków.
7. (2pkt) Niech $T = (V, E)$ będzie drzewem a $P(u, v)$ niech oznacza ścieżkę w T (rozumianą jako zbiór krawędzi) łączącą wierzchołki u i v.

   Ułóż algorytm, który dla drzewa T znajduje trzy wierzchołki a, b, c, dla których zbiór $\{e \in E : e \in P(a, b) \cup P(a, c) \cup P(b, c)\}$ jest maksymalnie duży.
8. (2pkt) Operacja $swap(i, j)$ na permutacji powoduje przestawienie elementów znajdujących się na pozycjach i oraz j. Koszt takiej operacji określamy na $|i - j|$. Kosztem ciągu operacji *swap* jest suma kosztów poszczególnych operacji.

   Ułóż algorytm, który dla danych π oraz σ - permutacji liczb $\{1, 2, \ldots, n\}$, znajdzie ciąg operacji *swap* o najmniejszym koszcie, który przekształca permutację π w permutację σ.
9. (1pkt) Na wykładzie przedstawiono zachłanny algorytm dla problemu *Pokrycia zbioru*, znajdujący rozwiązania, które są co najwyżej $\log n$ razy gorsze od rozwiązania optymalnego.

   Pokaż, że istnieją dane, dla których rozwiązania znajdowane przez ten algorytm są blisko $\log n$ gorsze od rozwiązań optymalnych.