Spamiętywanie i programowanie dynamiczne

1. Jaka jest minimalna liczba monet lub banknotów potrzebna do wydania nzł reszty, przyjmując, że w obrocie są dostępne monety o zadanych (w postaci tablicy) nominałach?
2. Na ile sposobów można wydać nzł reszty przyjmując, że w obrocie są dostępne monety o zadanych (w postaci listy) nominałach.
3. Dana jest kwota k oraz zestaw nominałów banknotów $b = \{b_1, b_2, \ldots, b_n\}$. Mamy dowolną liczbę banknotów każdego z nominałów. Napisz procedurę `int banknoty(int k, const vector<int> &b)`, która dla danych k i b obliczy minimalną liczbę różnych nominałów banknotów jakich należy użyć do wydania kwoty k. Jeżeli nie da się wydać kwoty k używając podanych nominałów, to procedura powinna zwrócić -1. Na przykład, `banknoty(49, {10, 15, 11, 12, 20}) == 3`. Liczbę 49 można uzyskać na dwa sposoby: $49 = 12 + 12 + 10 + 15 = 11 + 11 + 15 + 12$.
4. (*)[XII OI, zadanie *Banknoty*] W bankomacie znajdują się banknoty o nominałach $b_1, b_2, \ldots, b_n$, przy czym banknotów o nominale b_i jest w bankomacie c_i. Jak wypłacić zadaną kwotę używając jak najmniejszej liczby banknotów?
5. Dana jest tablica zawierająca dostępne nominały monet i banknotów. Tablica ta zawiera dodatnie liczby całkowite i jest uporządkowana rosnąco. Napisz procedurę `int unikat(const vector<int> & b, int k)`, która dla danej tablicy nominałów i nieujemnej kwoty k zwróci największą taką kwotę, która: nie przekracza k i może być wydana w dokładnie jeden sposób. Dwa sposoby wydania danej kwoty są różne, jeżeli mają różne multi-zbiory użytych nominałów.
6. Atomy bajtonium mają po k elektronów ($k > 0$). Elektrony w atomach bajtonium mogą znajdować się na n różnych powłokach, ponumerowanych od 0 do $n - 1$, przy czym na każdej powłoce może znajdować się co najwyżej jeden elektron. Masz daną tablicę t, posortowaną niemalejąco, zawierającą liczby całkowite z zakresu od 0 do n^2. Tablica ta opisuje energię elektronów znajdujących się na poszczególnych powłokach, to znaczy, elektron znajdujący się na powłoce i ma energię $t[i]$. Napisz procedurę `vector<int> energia(int k, const vector<int> &t, int e)`, która na podstawie liczby k, tablicy t oraz energii e wyznaczy orbity, na których powinny znajdować się elektrony w atomie bajtonium tak, żeby ich łączna energia wynosiła e. Wynikiem powinna być tablica k numerów powłok z elektronami, uporządkowana rosnąco. Jeżeli uzyskanie danej energii nie jest możliwe, wynikiem powinna być pusta tablica.
   Przykład: wynikiem `energia(4, {2, 8, 12, 12, 20}, 42)` powinno być `{0, 1, 2, 4}` lub `{0, 1, 3, 4}`, natomiast `energia(4, {2, 8, 12, 12, 20}, 60) == {}`.
7. (*) Antyczny bardzo długi kijek został połamany na kawałki. Należy go skleić. Wszystkie kawałki zostały już ułożone w odpowiedniej kolejności, ale nie wiadomo ile potrzeba kleju. Długości kolejnych kawałków są dane w postaci tablicy $\{x_1, x_2, \ldots, x_n\}$. Do slejenia kawałków długości a i b potrzeba $\max(a, b)$ ml kleju. Po ich sklejeniu otrzymujemy jeden kawałek długości $a + b$.
   Napisz procedurę `int klej(const vector<int> & x)`, która na podstawie długości kawałków obliczy minimalną ilość kleju (w ml) potrzebną do sklejenia wszystkich kawałków.
8. Dana jest tablica dodatnich liczb całkowitych $t = \{x_1, x_2, \ldots, x_n\}$. Spójny fragment tej tablicy $\{x_i, \ldots, x_j\}$ nazwiemy *segmentem Fibonacciego* rzędu k, jeżeli:
   - $i = j$ oraz $x_i = x_j = k$, lub
   - istnieje takie $l, i \leq l < j$, że $\{x_i, \ldots, x_l\}$ oraz $\{x_{l+1}, \ldots, x_j\}$ są segmentami Fibonacciego rzędu $k - 1$ i $k - 2$ (w dowolnej kolejności).

   Napisz procedurę `int segment(const vector<int> &t)`, która dla danej tablicy t wyznaczy największy rząd występującego w niej segmentu Fibonacciego. Jeżeli nie występuje w niej żaden taki segment, to poprawnym wynikiem jest zero.
   Przykład: `segment({1, 3, 2, 3, 4, 5, 3, 6}) == 7`. Segment Fibonacciego rzędu 7, to $\{3, 2, 3, 4, 5\}$ ($Fib_7 = 13$).
9. (*)[XII OI, zadanie *Autobus*] Ulice miasta tworzą szachownicę -- prowadzą z północy na południe, lub ze wschodu na zachód, a każda ulica prowadzi na przestrzał przez całe miasto -- każda ulica biegnąca z północy na południe krzyżuje się z każdą ulicą biegnącą ze wschodu na zachód i vice versa. Ulice prowadzące z północy na południe są ponumerowane od 1 do n, w kolejności z zachodu na wschód. Ulice prowadzące ze wschód na zachód są ponumerowane od 1 do m, w kolejności z południa na północ. Każde skrzyżowanie i-tej ulicy biegnącej z północy na południe i j-tej ulicy biegnącej ze wschodu na zachód oznaczamy parą liczb (i, j) (dla $1 \leq i \leq n, 1 \leq j \leq m$).
   Po ulicach miasta kursuje autobus. Zaczyna on trasę przy skrzyżowaniu $(1, 1)$, a kończy przy skrzyżowaniu (n, m). Ponadto autobus może jechać ulicami tylko w kierunku wschodnim i/lub północnym.
   Przy niektórych skrzyżowaniach znajdują się pasażerowie oczekujący na autobus. Rozmieszczenie pasażerów jest dane w postaci prostokątnej tablicy *a* $n \times m$ liczb całkowitych -- $a[i-1][j-1]$ oznacza liczbę pasażerów czekających przy

?

skrzyżowaniu (i, j).
Napisz procedurę `autobus`, która na podstawie tablicy *a* obliczy maksymalną liczbę pasażerów, których może zabrać autobus. Zakładamy, że w autobusie zmieści się dowolna liczba pasażerów.

10. [VII OI, zadanie Jajka, zmodyfikowane] Dostaliśmy k identycznych wytrzymałych strusich jaj. Okazuje się, że są one wytrzymałe na upadki z dużych wysokości. Mieszkamy w n-piętrowym wieżowcu. Chcemy sprawdzić, jakie jest maksymalne piętro p ($0 \leq p \leq n$), z którego jak zrzucimy strusie jajo, to się ono nie zbije.
Napisz funkcję `int liczba_prob(int n, int k)` taką, że $\texttt{liczba\_prob}(n, k)$ zwróci najmniejszą (pesymistycznie) liczbę prób zrzucania jaj potrzebną do wyznaczenia p mając do dyspozycji k jaj. Inaczej mówiąc, szukamy takiej strategii wyznaczenia p, która w najgorszym przypadku będzie wymagać jak najmniejszej liczby zrzucań jajek.
Uwagi:
    - Jajo zbije się tylko wtedy, gdy zrzucamy je z piętra wyższego niż p.
    - Jeżeli jajo się zbije, to je tracimy. Jeśli nie, to można je użyć do kolejnej próby.
    - Budynek ma $n + 1$ kondygnacji, wliczając parter.
    - Jeżeli jajo zrzucone z parteru też się zbija, to poprawnym wynikiem jest -1.

    Na przykład: $\texttt{liczba\_prob}(n, 1) == n + 1$, $\texttt{liczba\_prob}(4, 2) == 3$.
11. Mamy dany (w postaci wektora) ciąg dodatnich liczb całkowitych $\{x_1, x_2, \ldots, x_n\}$. Na ciągu tym można wykonywać operacje „sklejania", tzn. dwa kolejne elementy ciągu można zastąpić jednym, równym ich sumie. Napisz procedurę: `int sklejanie(const vector<int> & x)` obliczającą minimalną liczbę operacji „sklejania", które należy wykonać, aby ciąg był rosnący. Na przykład, `sklejanie({3, 5, 4, 2, 7}) == 1`.
Równoważne zadanie pojawiło się na konkursie USACO Open Gold w 2009 r. i zostało opisane w Delcie.
12. Na rozgałęzieniach drzewa siedzą wróble. Każde rozgałęzienie drzewa waży 1 kilogram, a nieujemna waga siedzących na tym rozgałęzieniu wróbli podana jest jako wartości w węzłach drzewa:
```
typedef struct node *bin_tree;
struct node {
    int val;
    bin_tree left, right;
};
```
Waga drzewa to suma wag rozgałęzień i siedzących na nich wróbli. Drzewo nazwiemy zrównoważonym, jeśli w każdym rozgałęzieniu waga dwóch poddrzew będzie taka sama. Rzucając celnie kamieniem w rozgałęzienie drzewa możemy przegonić wszystkie wróble z poddrzewa zaczepionego w tym rozgałęzieniu. Napisz funkcję `bool rownowaga(bin_tree t)`, która dla danego drzewa określa, czy da się tak rzucać kamieniami w drzewo, aby stało się ono zrównoważone. Podaj złożoność czasową i pamięciową swojego rozwiązania.
Dla przykładu, aby zrównoważyć następujące drzewo: `node(3, node(5, node (1), node(7)), node(2))` wystarczy raz rzucić kamieniem (w jego lewe poddrzewo).
13. Latarnie oświetlają prostą alejkę w parku. Każda latarnia jest opisana przez parę postaci $((x, y), p)$ oznaczającą, że latarnia może oświetlić (domknięty) odcinek alejki od x do y i pobiera wtedy pW energii elektrycznej ($x < y, p > 0$). Dana jest tablica opisująca latarnie, oraz para liczb (a, b), $a < b$, opisująca odcinek alejki, który chcemy oświetlić. Napisz procedurę `int moc(const vector<pair<pair<int, int>, int>> & t, int a, int b)`, która obliczy minimalną moc potrzebną do oświetlenia danego odcinka alejki. Jeżeli jego oświetlenie nie jest możliwe, to poprawnym wynikiem jest -1. Przykład:
`moc({{(2, 5), 10), ((10, 12), 4), ((-1, 2), 8), ((6, 9), 10), ((9, 15), 20), ((4, 7), 8), ((0, 12), 100), ((8, 10), 4), ((-2, 2), 6)}, 1, 11) == 42`
14. [XVI OI, zadanie *Konduktor*] Pociąg na trasie zatrzymuje się na n stacjach, ponumerowanych od 0 do $n - 1$. Dana jest tablica a, taka że $a[i][j]$ (dla $0 \leq i < j < n$) jest równe liczbie pasażerów, którzy wsiadają na stacji i i wysiadają na stacji j. Możesz wybrać k momentów (między kolejnymi stacjami), kiedy konduktor sprawdza bilety. Oblicz, jaka jest maksymalna liczba pasażerów, którym zostaną sprawdzone bilety.
15. [BOI'2003, przeformułowane] Tomcio chce zrobić model drzewa (acyklicznego nieskierowanego grafu spójnego) z nitki i szklanych koralików i to taki, w którym koraliki tego samego koloru nie sąsiadują ze sobą. Nitkę już ma, ale koraliki musi kupić za swoje kieszonkowe. Ceny koralików różnych kolorów są różne, dla każdego dodatniego całkowitego n w sprzedaży są koraliki jednego koloru w cenie n gr za koralik. Tomcio zastanawia się z jakich koralików zrobić model drzewa, tak aby wydać jak najmniej.
Opis drzewa ma postać struktury wskaźnikowej, w której każdy węzeł zawiera wektor wskaźników do jego synów. Napisz procedurę `koraliki`, która na podstawie opisu drzewa obliczy minimalny koszt koralików potrzebnych do wykonania jego modelu.
Uwaga: Dwa kolory wystarczą do wykonania modelu, ale nie dają minimalnego kosztu. Możesz założyć, że optymalna liczba kolorów nie przekracza $\log_2 n$, gdzie n to liczba koralików.
16. (*)[IX OI, zadanie „Waga"] Dany jest zestaw odważników. Czy można na szalkach wagi położyć część odważników tak, żeby waga nadal była zrównoważona? Jeśli tak, to jaki najcięższy odważnik można przy tym użyć?
17. (*)[IX OI, zadanie „Nawiasy"] Dane jest wyrażenie postaci $x_1 \pm x_2 \pm \cdots \pm x_n$. Na ile różnych sposobów można uzyskać wyrażenie równoważne danemu wstawiając nawiasy do wyrażenia $x_1 - x_2 - \cdots - x_n$? Uwaga: wstawiamy $n - 1$ par

nawiasów w pełni określając kolejność wykonywania odejmowań.

18. [XI OI, zadanie „Sznurki"] Interesują nas modele drzew (spójnych grafów acyklicznych) wykonane ze sznurka. Każde drzewo składa się z wierzchołków i pewnej liczby krawędzi łączących różne wierzchołki. Ten sam wierzchołek może być połączony z wieloma innymi wierzchołkami. Wierzchołki grafów mają być modelowane przez węzły zawiązane na kawałkach sznurka, a krawędzie przez odcinki sznurka pomiędzy węzłami. Każdy węzeł może być wynikiem zasuplenia kawałka sznurka lub związania w tym samym miejscu wielu kawałków. Każda krawędź ma długość 1. Długość sznurka użytego do zrobienia węzłów jest pomijalna.
    Chcemy zoptymalizować liczbę i długość kawałków sznurka użytych do wykonania modelu drzewa:
    - w pierwszym rzędzie należy zminimalizować liczbę potrzebnych kawałków sznurka,
    - zakładając, że liczba kawałków sznurka jest minimalna, należy zminimalizować długość najdłuższego kawałka sznurka.

Last modified: Saturday, 21 January 2023, 12:32 PM