Programowanie dynamiczne, zachłanne i back-tracking -- powtórzenie

Rozwiązanie każdego z poniższych zadań wymaga użycia jednej z trzech technik: back-trackingu, programowania dynamicznego lub zachłannego. Pytanie której?

1. [PCh, Zadanie o misjonarzach i kanibalach] Przez rzekę chcą przeprawić się kanibale i misjonarze. Kanibali i misjonarzy jest po trzech. Mają jedną łódkę, którą może płynąć do dwóch osób. Łódką umieją wiosłować kanibale i tylko jeden misjonarz. Jeżeli w dowolnej chwili, na dowolnym brzegu rzeki będzie więcej kanibali, niż misjonarzy, to zjedzą oni misjonarzy. W jaki sposób wszyscy mogą bezpiecznie przeprawić się na drugi brzeg.
2. [CLR] Minimalny koszt mnożenia *n* macierzy, o wymiarach $x_1 \times x_2, x_2 \times x_3, \ldots, x_n \times x_{n+1}$.
3. [CLR 17.2-5] Dany jest zbiór punktów na osi. Podaj minimalną liczbę odcinków jednostkowych, jakimi można pokryć punkty z tego zbioru.
4. Rozważmy taką oto łamigłówkę: Dany jest ciąg liczb całkowitych $t = [x_1; x_2; \ldots; x_n]$. W jednym ruchu możemy wybrać dowolne dwa równe elementy, usunąć je i wstawić na koniec ciągu ich sumę. Takie ruchy możemy powtarzać. Jaka jest minimalna długość listy jaką można uzyskać?
   Napisz procedurę `int lamiglowka(vector<int> t)`, która dla danego ciągu t wyznaczy minimalną długość ciągu jaką można uzyskać.
   Na przykład, `lamiglowka({4, 3, 4, 5, 12, 2, 3, 1, 6, 1}) == 4`.
5. Rozważmy taką oto łamigłówkę: Dany jest ciąg liczb całkowitych $t = [x_1; x_2; \ldots; x_n]$. W jednym ruchu możemy wybrać dowolne dwa różne elementy, usunąć je i wstawić na koniec ciągu wartość bezwzględną ich różnicy. Takie ruchy możemy powtarzać. Kończymy, gdy wszystkie elementy ciągu są sobie równe. Jaka jest minimalna wartość, jaką można na koniec uzyskać w ciągu?
   Napisz procedurę `int lamiglowka(vector<int> t)`, która dla danego ciągu t wyznaczy minimalną wartość jaką można uzyskać.
6. W korytarzu harcują myszy. Korytarz ma długość n metrów. Dana jest tablica n nieujemnych liczb całkowitych a opisująca gdzie jest ile myszy, dokładnie na odcinku od i do $i+1$ metrów od wejścia do korytarza jest $a[i]$ myszy.
   Dysponujesz k kotami ($k \leq n$). Twoim zadaniem jest takie rozmieszczenie kotów w korytarzu, żeby złapały jak najwięcej myszy. Każdy kot może pilnować ustalonego przez Ciebie spójnego fragmentu korytarza (na przykład, może mieć założoną smycz odpowiedniej długości, przymocowaną do podłogi pośrodku pilnowanego fragmentu korytarza). Fragmenty korytarza pilnowane przez różne koty nie mogą zachodzić na siebie (żeby koty się nie pobiły a smycze nie poplątały), choć mogą się stykać. Niektóre fragmenty korytarza mogą pozostać niepilnowane przez żadnego kota.
   Kot, który pilnuje fragmentu korytarza od i do j metrów od wejścia (dla $i < j$), na którym znajduje się $s = a[i] + a[i+1] + \cdots + a[j-1]$ myszy, złapie: $\max(s - (j-i-1)^2, 0)$ myszy.
   Na przykład, dla $k = 2$ oraz $a = \{1, 5, 1, 4, 3, 2, 7, 0\}$ poprawnym wynikiem jest 14, jeden kot może pilnować fragmentu korytarza od 1 do 4 metrów od wejścia (łapiąc 6 z 10 myszy), a drugi może pilnować fragmentu korytarza od 4 do 7 metrów od wejścia (łapiąc 8 z 12 myszy).
   Napisz procedurę `int myszy(int k, vector<int> a)`, która dla danej liczby $k \geq 0$ oraz tablicy a, wyznaczy maksymalną liczbę myszy, jakie złapią koty.
   Rozwiązanie tego zadania zostało omówione w Delcie
7. Rozpatrujemy przedstawienia liczby naturalnej n jako sumy różnych składników nieparzystych, tzn. rozważamy takie ciągi $(x_1, \ldots, x_k)$, że:
   - $1 \leq x_i$ i x_i jest nieparzyste (dla $i = 1, \ldots, k$),
   - $x_1 > x_2 > \cdots > x_k$,
   - $x_1 + \cdots + x_k = n$.
   1. Napisz procedurę, która dla danej liczby n obliczy, na ile różnych sposobów można ją przedstawić jako sumę różnych nieparzystych składników. Na przykład, dla $n = 2$ poprawnym wynikiem jest 0, a dla $n = 12$ poprawnym wynikiem jest 3.
   2. Napisz procedurę, która dla danej liczby n wyznaczy (pewne) przedstawienie tej liczby jako sumy maksymalnej liczby nieparzystych składników. Na przykład, dla $n = 42$ poprawnym wynikiem może być `{15, 11, 7, 5, 3, 1}`.
8. Rozważamy ciągi liczb całkowitych $(x_1, x_2, \ldots, x_k)$ konstruowane zgodnie z następującymi regułami:
   - $x_1 = 1$,
   - $x_{i+1} = x_i + 1$, lub

?

- $x_{i+1} = x_i \cdot f_i$ dla pewnej liczby Fibonacciego f_i.

Napisz procedurę `vector<int> ciag(int n)`, która dla danej dodatniej liczby całkowitej n wyznaczy najkrótszy taki ciąg $(x_1, x_2, \ldots, x_k)$, który spełnia powyższe warunki, oraz $x_k = n$.
Na przykład `ciag(42)` powinno zwrócić $\{1, 2, 42\}$ lub $\{1, 21, 42\}$.

9. Rozważamy ciągi liczb całkowitych $(x_1, x_2, \ldots, x_k)$ konstruowane zgodnie z następującymi regułami:
   - $x_0 = 1$,
   - dla każdego $k > 0$ wybieramy pewne $0 \leq i, j < k$ i przyjmujemy $x_k = x_i + x_j$.

   Napisz procedurę `vector<int> ciag(int n)`, która dla danej dodatniej liczby całkowitej n wyznaczy najkrótszy taki ciąg $(x_1, x_2, \ldots, x_k)$, który spełnia powyższe warunki, oraz $x_k = n$.
10. Rozważamy ciągi liczb całkowitych $(x_1, x_2, \ldots, x_k)$ konstruowane zgodnie z następującymi regułami:
    - $x_1 = 1$,
    - $x_{i+1} = x_i + 1$, lub
    - $x_{i+1} = x_i \cdot p_i$ dla pewnej liczby pierwszej p_i.

    Napisz procedurę `vector<int> ciag(int n)`, która dla danej dodatniej liczby całkowitej n wyznaczy najkrótszy taki ciąg $(x_1, x_2, \ldots, x_k)$, który spełnia powyższe warunki, oraz $x_k = n$.
    Możesz założyć, że dane są procedury:
    - `bool prime(int k)` sprawdzająca czy dana liczba jest pierwsza, oraz
    - `vector<int> primes(int n)`, która dla danej dodatniej liczby całkowitej n zwraca listę liczb pierwszych nie przekraczających n (uporządkowaną rosnąco).
11. Dla danych dodatnich liczb całkowitych l i m szukamy przedstawienia ułamka $\frac{l}{m}$, jako możliwie najkrótszej sumy odwrotności dodatnich liczb całkowitych:

$$\frac{l}{m} = \frac{1}{s_1} + \frac{1}{s_2} + \cdots + \frac{1}{s_k}$$

    Napisz procedurę `vector<int> rozklad(int l, int m)`, która dla danych liczb l i m wyznaczy szukany ciąg mianowników $\{s_1, \ldots, s_k\}$.
12. Dana jest dodatnia liczba całkowita x. Interesują nas różne rozkłady x na sumę różnych (dodatnich) liczb Fibonacciego. Dwa rozkłady, które różnią się jedynie kolejnością składników uznajemy za takie same. Tylko rozkłady, które mają inne zbiory składników uznajemy za różne.
    Napisz procedurę `int rozklady(int x)`, która dla danej liczby x policzy na ile różnych sposobów można przedstawić x jako sumę różnych (dodatnich) liczb Fibonacciego.
    Na przykład `rozklady(42) == 6`.
13. Dany jest ciąg dodatnich liczb całkowitych $\{x_1, x_2, \ldots, x_n\}$ oraz dodatnia liczba całkowita k. Napisz procedurę `vector<vector<int>> podzial(vector<int> x, int k)`, która podzieli dany ciąg na k spójnych fragmentów $\{\{x_1, \ldots, x_{n_1}\}, \{x_{n_1+1}, \ldots, x_{n_2}\}, \ldots, \{x_{n_{k-1}+1}, \ldots, x_n\}\}$ w taki sposób, że $\max\{x_1 + \cdots + x_{n_1}, x_{n_1+1} + \cdots + x_{n_2}, \ldots, x_{n_{k-1}+1} + \cdots + x_n\}$ jest jak najmniejsze. Jeżeli $k > n$, to w liście wynikowej mogą występować puste listy.
    Na przykład `podzial({1, 4, 2, 2, 4, 1, 2, 4}, 3)` może się równać `{{1, 4, 2}, {2, 4}, {1, 2, 4}}`.
14. Na przystanku autobusowym pojawiają się pasażerowie czekający na autobus. O określonych godzinach podjeżdżają puste autobusy, do których mogą wsiadać pasażerowie. Należy obliczyć minimalną wielkość autobusów, pozwalającą na przewiezienie wszystkich pasażerów (wszystkie autobusy są tej samej wielkości). Tablica p zawierająca n nieujemnych liczb całkowitych opisuje pojawianie się pasażerów, w chwili i na przystanek przychodzi $p[i]$ pasażerów. Tablica a opisuje momenty przyjazdu autobusów na przystanek -- jest ona uporządkowana niemalejąco i zawiera liczby całkowite z zakresu od 0 do n. Napisz procedurę `int autobusy(vector<int> p, vector<int> a)`, która oblicza minimalną konieczną wielkość autobusów. Jeżeli nie da się obsłużyć wszystkich pasażerów, to poprawnym wynikiem jest -1.
    Na przykład, `autobusy({1, 0, 2, 1, 2, 0, 1, 2, 6, 0, 6, 0}, {2, 6, 8, 11}) == 7`.
15. [XVI OI, zadanie Słonie] Dana jest tablica p długości n zawierająca permutację liczb ze zbioru $\{0, \ldots, n-1\}$. Chcemy posortować tablicę p rosnąco, ale jedyną operacją modyfikującą, którą możemy wykonywać na niej jest zamiana zawartościami dwóch komórek o podanych indeksach. Zamiana komórek zawierających liczby x i y kosztuje $x + y + 1$.
    Napisz procedurę `int koszt(vector<int> p)`, która oblicza minimalny koszt posortowania tablicy p.
16. [II OI] Dany jest zestaw n czynności. Wykonanie każdej z tych czynności trwa tym dłużej, im później się ją zacznie, konkretnie jeżeli zaczniemy wykonywać czynność i w chwili t, to jej wykonanie będzie trwać $a_i t + b_i$. Jaki jest minimalny czas wykonania wszystkich tych czynności, jeżeli zaczynamy je wykonywać w chwili 0?
17. Rozważamy drzewa postaci:

```
typedef struct node *expr;
struct node {
    int val;
    bin_tree left, right;
```

```
};
const int NWD = -2;
const int NWW = -3;
```

Drzewa takie reprezentują wyrażenia. Jeśli `val == NWD`, to oznacza największy wspólny dzielnik dwóch podwyrażeń. Jeśli `val == NWW`, to oznacza najmniejszą wspólną wielokrotność dwóch podwyrażeń. W obu tych przypadkach, wskaźniki do poddrzew (reprezentujących podwyrażenia) nie mogą być puste. Jeśli `val >e; 0`, to oba wskaźniki do poddrzew muszą być puste. `val > 0` reprezentuje liczbę równą `val`. `val == 0` oznacza puste miejsce, w które należy wstawić dodatnią liczbę.

Napisz procedurę `int wstaw(expr t, int n)`, która dla danego drzewa t i dodatniej liczby całkowitej n znajdzie taką najmniejszą dodatnią liczbę całkowitą x, że gdy wstawimy x w miejsce **wszystkich** wystąpień $\texttt{val} = 0$ w t, to wartością wyrażenia będzie n. Jeżeli taka liczba x nie istnieje, to poprawnym wynikiem jest 0.

Na przykład, dla $t = \texttt{NWW(NWW(2, 0), NWD(0, 49))}$ i $n = 42$, poprawnym wynikiem jest 21.

18. Przez park biegnie długa prosta alejka. Przy tej alejce, w określonych miejscach, znajduje się n ławeczek. Wzdłuż alejki chcemy zamontować k latarni. Dla każdej ławeczki interesuje nas odległość do najbliżej położonej latarni. Latarnie chcemy rozmieścić tak, żeby maksymalna odległość od ławeczki do najbliżej niej położonej latarni była jak najmniejsza, przy czym latarnie mogą być umieszczone tylko w miejscach o współrzędnych całkowitych. Masz daną posortowaną tablicę n liczb całkowitych określających położenie ławeczek oraz liczbę k. Napisz procedurę `vector<int> latarnie(vector<int> t, int k)`, która na ich podstawie wyznaczy gdzie należy postawić latarnie i zwróci posortowaną tablicę z ich pozycjami.
Na przykład, dla tablicy $\{-4, -3, -1, 2, 2, 3, 8, 10, 11\}$ oraz $k = 3$, jednym z możliwych wyników jest $\{-2, 2, 10\}$.
Żadna z ławeczek nie jest oddalona o więcej niż 2 od którejś z latarni.
19. Dane jest drzewo typu: `typedef struct node *bin_tree;`

```
struct node {
    int val;
    bin_tree left, right;
};
```

opisujące kształt choinki.

1. [PCh] W węzłach choinki możemy umieszczać świeczki. Powiemy, że choinka jest oświetlona, jeżeli dla każdej krawędzi choinki (niepustego wskaźnika) przynajmniej w jednym jej końcu znajduje się świeczka. Napisz procedurę, która dla danej choinki obliczy minimalną liczbę świeczek potrzebnych do oświetlenia choinki.
2. [PCh] Na każdej krawędzi (niepusty wskaźnik) choinki wieszamy jedną bombkę. Mamy do dyspozycji trzy rodzaje bombek: czerwone po 2 zł, niebieskie po 5 zł i srebrne po 7 zł. Choinkę należy przystroić w taki sposób, żeby na krawędziach o końcach w tym samym węźle wisiały bombki o różnych kolorach, a łączny koszt przystrojenia całej choinki był jak najmniejszy.
3. W węzłach choinki wieszamy bombki. Dostępne są trzy rodzaje bombek:
   - czerwone, wykonane z miedzi, każda taka bombka waży 1kg,
   - srebrne, wykonane z białego złota niskiej próby, każda taka bombka waży 2kg,
   - złote, wykonane ze szczerego złota, każda taka bombka waży 3kg.

   Bombki należy zawiesić tak, aby choinka nie chwiała się, tzn. dla każdego rozgałęzienia (węzła, z którego wychodzą dwa niepuste wskaźniki) łączny ciężar bombek zawieszonych po lewej i po prawej stronie musi być taki sam. Pytanie czy jest to możliwe, a jeżeli tak, to na ile sposobów?
   Napisz procedurę, która dla danej choinki obliczy na ile sposobów można powiesić na niej bombki.

Last modified: Saturday, 21 January 2023, 2:36 PM