ASD - ćwiczenia III

Dowodzenie poprawności programów iteracyjnych

Nieformalnie o poprawności programów:

- poprawność częściowa – jeżeli program zakończy działanie dla danych wejściowych spełniających założony warunek wstępny {WP}, to wynik jest zgodny z założonym warunkiem końcowym {WK},
- poprawność całkowita – jeżeli dane wejściowe spełniają założony warunek wstępny {WP}, to program kończy działanie z wynikiem zgodnym z założonym warunkiem końcowym {WK},
- **wniosek:** poprawność całkowita = poprawność częściowa + dowód skończoności wykonania programu.

Poprawność częściowa programów iteracyjnych:

- warunek początkowy {WP} – co zakładamy o zmiennych użytych w pętli iteracyjnej,
- warunek końcowy {WK} – co zakładamy o rezultacie pętli iteracyjnej,
- niezmiennik {NZ} – formuła logiczna, która definiuje relacje między zmiennymi użytymi w pętli iteracyjnej, własności niezmiennika:
  - niezmiennik jest zachowany przed pętlą iteracyjną,
  - niezmiennik nie musi być zachowany w trakcie wykonywania pętli iteracyjnej,
  - niezmiennik „odtwarza się“ i ponownie jest zachowany na końcu pętli iteracyjnej,
  - niezmiennik musi być „wystarczająco silny“, by można było na jego podstawie wnioskować o poprawności działania pętli iteracyjnej,
- wnioskowanie o poprawności pętli iteracyjnej:

$$\neg\{\mathtt{DZ}\}\wedge\{\mathtt{NZ}\}\Rightarrow\{\mathtt{WK}\},$$

  gdzie {DZ} jest dozorem pętli.

Zadania

1. Udowodnij poprawność poniższej funkcji SILNIA() względem niezmiennika {NZ: s = i!}, warunku początkowego {WP: n ∈ ℕ} oraz warunku końcowego {WK: s = n!}.

```
int SILNIA(int n) {
  {WP: n ∈ ℕ}
  int s:=1, i:=0;
  {NZ: s = i!}
  while (i < n) do
    i:=i+1;
    s:=s*i;
  od;
  {WK: s = n!}
  return s;
}
```

2. Dla podanej poniżej funkcji `FUNCTION()`, warunku początkowego {WP: $a \in \mathbb{N} \wedge b \in \mathbb{N}$} oraz niezmiennika pętli {NZ: $s \cdot p^w = a^b$} ustal warunek końcowy {WK}. Co jest wynikiem działania tej funkcji? Jaka jest jej złożoność czasowa względem liczb a i b?

```
int FUNCTION(int a, int b) {
  {WP: a ∈ ℕ, b ∈ ℕ}
  int s:=1, p:=a, w:=b;
  {NZ: s · p^w = a^b}
  while (w > 0) do
    if (w mod 2 = 0) then
      p:=p*p;
      w:=w/2;
    else
      s:=s*p;
      w:=w-1;
    fi;
  od;
  (WK: ...}
  return s;
}
```

3. Dla poniższej funkcji `FUNCTION()`, warunku początkowego {WP: $a \in \mathbb{N} \setminus \{0\}$} ustal niezmiennik pętli {NZ} oraz warunek końcowy {WK}. Jaka jest złożoność tej funkcji względem:
   (a) wartości liczby a,
   (b) rozmiaru (wyrażonego liczbą niezbędnych bitów zapisu) liczby a?

```
int FUNCTION(int a) {
  {WP: a ∈ ℕ \ {0}}
  int i:=a, j:=a;
  {NZ: ...  }
  while (j > 1) do
    j:=j div 3;
    i:=i+1;
  od;
  {WK: ...  }
  return i;
}
```

Zadanie o pracownikach i synie szefa

Przedsiębiorstwo zatrudnia $n > 10^3$ pracowników etatowych $W[1], W[2], \ldots, W[n]$, z których każdy rocznie wypracowuje na rzecz firmy $I[i]$ złotych dochodu (dla ułatwienia przyjmujemy, że $\forall i, j \in \{1, 2, \ldots, n\} \wedge i \neq j \; : \; I[i] \neq I[j]$). Prezes tego przedsiębiorstwa pragnie gratyfikować k pracowników, których łączny roczny dochód jest największy spośród wszystkich możliwych k elementowych sum dochodów pracowników $W[1], W[2], \ldots, W[n]$:

- zaproponuj algorytm $\mathcal{A}$ wspomagający prezesa w wyznaczeniu k elementowej grupy pracowników którym należy się gratyfikacja, jeżeli do dyspozycji dostajesz uporządkowaną alfabetycznie listę nazwisk pracowników wraz z wypracowanym rocznym dochodem. Zadbaj o to, aby spełnione były zależności $A(\mathcal{A}, n) = O(kn)$ oraz $W(\mathcal{A}, n) = O(kn^2)$,

- w trakcie rozmowy z sekretarką prezes dowiedział się, że jego syn znalazł się w grupie trzech najlepiej zarabiających pracowników i zaraz zabrał się za weryfikację tej wiadomości na podstawie wcześniej opisanej listy pracowników. Niestety po dokładnie $n + \frac{n}{10}$ porównaniach dochodów $I[i]$ wybranych pracowników w gabinecie prezesa zadzwonił telefon. Czy rozpoczynając rozmowę telefoniczną prezes może być pewien, które miejsce w grupie trzech liderów zajmuje jego syn (jeżeli uważasz, że tak to podaj odpowiedni algorytm postępowania, jeżeli uważasz, że nie to podaj uzasadnienie dlaczego określona liczba porównań jest niewystarczająca)?

Zadanie o przeglądaniu nieskończonej struktury danych

Wyobraźmy sobie „nieskończoną" strukturę danych S, zaimplementowaną przy pomocy niezdefiniowanej struktury danych

```
typedef str_Structure [] Structure;

struct Structure {
  int number;
  ...
};
```

przechowującą liczby całkowite. Jest ona „wyposażona" w operację $GET(S, i)$, gdzie $i \in \mathbb{N} \setminus \{0\}$, może być dowolnie dużą liczbą naturalną. Operacja $GET(S, i)$ zwraca liczbę całkowitą przechowywaną w strukturze S pod indeksem i-tym, przy czym:

- pod każdym indeksem w rozważanej strukturze danych znajduje się dokładnie jedna liczba całkowita,
- przechowywane liczby są posortowane w porządku niemalejącym, czyli jeżeli $i < j$ to $GET(S, i) < GET(S, j)$,
- pierwsza liczba przechowywana w strukturze danych jest ujemna (tj. $GET(S, 1) < 0$),
- istnieje skończona liczba naturalna n taka, że $GET(S, n) > 0$,
- operacja $GET(S, i)$ ma koszt $O(1)$ dla każdego $i \in \mathbb{N} \setminus \{0\}$.

Opracuj funkcję

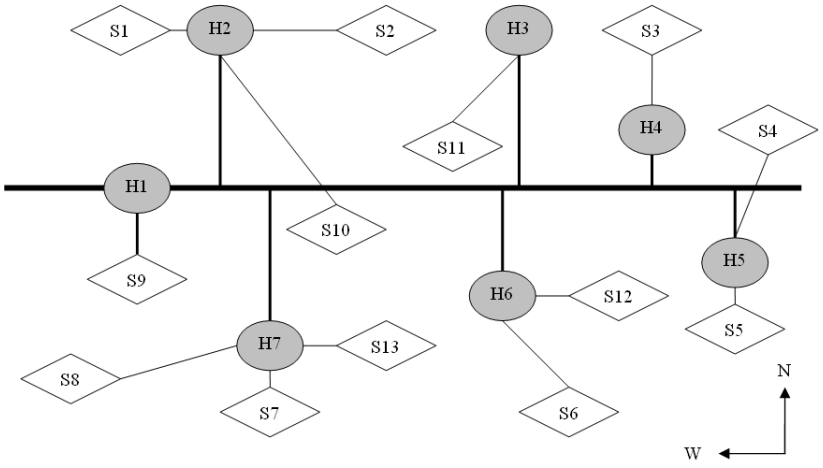
`int FIND(Structure S)`,

możliwie asymptotycznie najszybszą, która odnajdzie najmniejszy indeks w strukturze S, pod którym znajduje się liczba dodatnia (tj. minimalne i takie, że $GET(S, i) > 0$). Zakładamy, że dostęp do rozważanej struktury danych może odbywać się jedynie za pomocą zdefiniowanej operacji GET.

- Podaj słowny opis procedury `FIND()`.
- Napisz pseudokod procedury `FIND()` (w kodzie procedury można używać dowolnych algorytmów przedstawionych na wykładzie, bez potrzeby wypisywania ich implementacji, należy jednak wyjaśnić jak i do czego się ich używa).
- Oszacuj złożoność czasową algorytmu, przy czym dokładnie opisz składniki funkcji złożoności, jeżeli procedura składa się z kilku faz. Nie zapomnij precyzyjnie określić co stanowi argument funkcji złożoności.

Zadanie o najtańszej sieci komputerowej (do domu)

Pewna firma zamierza zbudować sieć komputerową $\mathcal{N}$, która ma połączyć wszystkie m stacji roboczych $S[1], S[2], \ldots, S[m]$ znajdujących się w jej siedzibie. Firma zajmuje całe pojedyncze piętro w jednym z warszawskich wieżowców. Każda stacja robocza ma być podłączona do jednego z n węzłów sieci $H[1], H[2], \ldots, H[n]$, gdzie $n < m$ i $n = 2k - 1$ dla $k \in \mathbb{N} \setminus \{0\}$. Kierownictwo firmy ustaliło już dla każdej stacji roboczej $S[i]$ indeks num węzła $H[num]$, do którego ma być przyłączona kablem sieciowym o minimalnej długości. Węzły sieci mają być podłączone do kabla głównego (także w najkrótszy możliwy sposób), który ma zostać położony w linii prostej z kierunku zachodniego w stronę wschodnią i ma przebiegać bezpośrednio przez dokładnie jeden węzeł. Zakładamy, że zarówno współrzędne zachodnie *west* oraz północne *north* każdego elementu sieci (tj. stacji roboczej, węzła) są z góry znane, zawierają się w zbiorze $\mathbb{Z}$ liczb całkowitych. Dodatkowo przyjmujemy, że dowolne dwa węzły sieci $H[i]$, $H[j]$ nie leżą na tej samej współrzędnej szerokości jak i długości geograficznej.



Rysunek 1: Przykład planowanej sieci komputerowej $\mathcal{N}$, dla $m = 13$ i $n = 7$.

Niech dalej poniższe struktury danych `WStation` oraz `Node` reprezentują kolejno stację roboczą oraz węzeł sieci komputerowej:

```
typedef str_WStation WStation;
typedef str_Node Node;

struct str_WStation {
  int west, north, num;
};

struct str_Node {
  int west, north;
};
```

Zaprojektuj funkcję

```
int FIND_BEST(WStation S[], Node H[], int k, int n),
```

która wyznaczy numer węzła sieci komputerowej przez który powinien przechodzić kabel główny, tak aby łączna suma długości wszystkich kabli łączących stacje robocze z węzłami i

węzły z kablem głównym była minimalna z możliwych – czyli koszty budowy sieci $\mathcal{N}$ będą minimalne :)). Długość kabla głównego pomijamy i zakładamy, że dowolne dwa punkty sieci można zawsze połączyć kablem sieciowym ułożonym w linii prostej.