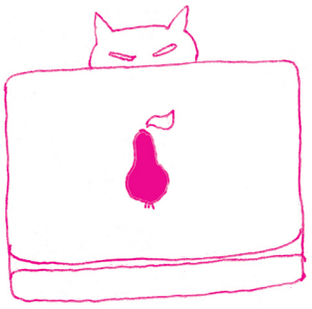


Poza samym zaprezentowanym (dość szybkim i eleganckim) algorytmem chciałbym zwrócić uwagę Czytelnika na pewną ideę, która za nim stoi. Myślę tu o idei abstrakcyjnych struktur danych. Przykładem takiej struktury jest właśnie kopiec. Z lotu ptaka o kopcu (tablicy kopcowej) można myśleć jak o takim wirtualnym czarnym pudełku, o zawartości którego wcale nie musimy zbyt wiele wiedzieć. Pomyślmy o sytuacji, w której nie chciało nam się analizować działania procedur `insert` oraz `deletemax`. Wiemy tylko, że pierwsza z nich dokłada element do czarnego pudełka, a druga – wyciąga największy. Dodatkowo poinformowano nas, że obie działają szybko (w czasie proporcjonalnym do logarytmu z liczby elementów w pudełku). To już wystarczy do zrealizowania nie tylko sortowania w czasie $O(n \log n)$, ale i napisania np. programu do obsługi harmonogramu zadań, w którym nieustannie pojawiają się nowe pozycje, z różnymi priorytetami, a komputer decyduje, czym się w danej chwili zająć. Takie myślenie (zapominamy o wnętrzu struktury danych i tylko patrzymy na jej funkcjonalność) nazywamy *abstrahowaniem* i ono jest absolutnie solą całej algorytmiki. Nie bez powodu większość kursów i podręczników algorytmiki nosi tytuł „Algorytmy i struktury danych”.

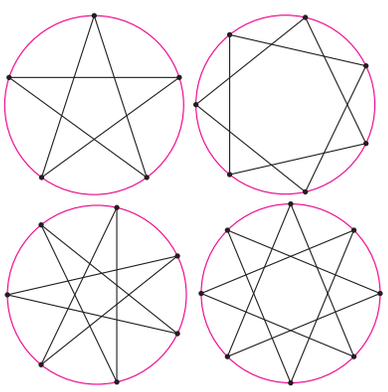
Świat abstrakcyjnych struktur danych to nie tylko kopce, ale i stosy, drzewa czerwono-czarne, kolejki czy miotły. Piękne nie tylko z nazwy, ale i dzięki pomysłowym implementacjom.

```
procedure sort;
var j : integer;
begin
   m := 1;
   for j := 2 to n do insert(a[j]); {faza 1}
   for j := n downto 2 do deletemax; {faza 2}
end;
```

```
procedure insert(v: integer);
var k, p : integer;
begin
   m := m + 1;
   a[m] := v;
   k := m div 2;
   p := m;
   while (k > 0) and (a[k] < v) do
   begin
      a[p] := a[k];
      p := k;
      k := k div 2;
      a[p] := v;
   end;
end;
```

```
procedure deletemax;
var k, p, r : integer;
begin
   r := a[m];
   a[m] := a[1];
   a[1] := r;
   m := m - 1;
   k := 1;
   while (k <= m div 2) do
   begin
      p := 2 * k;
      if (p < m) then
         if a[p] < a[p + 1] then p := p + 1;
      if r >= a[p] then break;
      a[k] := a[p];
      k := p;
   end;
   a[k] := r;
end;
```

Dziwna liczba gwiazdek foremnych



Gwiazdka foremna to łamana zamknięta, wpisana w okrąg i złożona z jednakowej długości cięciw, ale niebędąca wielokątem foremnym. Nie trzeba długo się zastanawiać, by stwierdzić, że odcinki takich łamanych muszą się przecinać.

Każdy łatwo sprawdzi, że obok narysowane są wszystkie gwiazdki mające nie więcej niż 8 wierzchołków.

Jeśli lista wszystkich dzielników pierwszych liczby n to $p_1, p_2, \ldots, p_k$ (uwaga, na liście nie ma powtórzeń, np. 64 ma listę złożoną z jednej liczby), to liczba gwiazdek foremnych mających n wierzchołków dana jest wzorem

$$\frac{n}{2} \cdot \left(\frac{p_1 - 1}{p_1}\right) \cdot \left(\frac{p_2 - 1}{p_2}\right) \cdot \ldots \cdot \left(\frac{p_k - 1}{p_k}\right) - 1,$$

Ale dlaczego akurat tyle?

M.K.