1. Porównaj pod kątem złożoności metodę sortowania przez wstawianie realizowane w tablicy z porządkowaniem ciągu danych przez dodawanie ich w odpowiednich miejscach listy jednokierunkowej (z wartownikiem lub bez). Opisz w szczególności czy (a jeśli tak, to w jakim stopniu) zastosowanie listy może zmniejszyć liczbę porównań i/lub podstawień niezbędnych do posortowania danych. Przeanalizuj w tym celu najgorszy przypadek.

2. Sortowanie przez łączenie, tzw. *merge sort*, polega na podziale ciągu danych na dwie równe części (jeśli `n` jest nieparzyste, pierwsza zawiera o jeden element więcej), rekurencyjnym wykonaniu tego sortowania dla każdej z nich, a następnie połączeniu posortowanych połówek w jeden uporządkowany ciąg. Rekursja kończy się, gdy przekazana jej część zawiera nie więcej niż 2 liczby i porządkowanie staje się trywialne.

Mając dwie funkcje (nie trzeba pisać ich kodu):

- `podziel(&P, &Q)`, która dzieli listę `P` na dwie równe części, pierwszą wskazywaną przez `P`, a drugą przez `Q` (`P` zawiera minimalnie 1 element, natomiast `Q` może być pusta);
- `połącz(&P, &Q)` tworzy z uporządkowanych list `P` i `Q` jedną listę uporządkowaną, która po wyjściu z tej funkcji wskazywana jest przez `P`, natomiast `Q` jest pusta (tak jak w jednym z zadań domowych).

Napisz kod funkcji `merge_sort(&P)` porządkującej listę `P`, która korzysta z wywołań funkcji `podziel` i `połącz`.

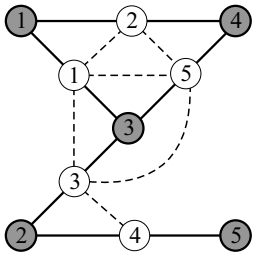
3. Graf G, którego wierzchołki etykietowane są literami `a, b, c,...` opisany jest przez następujące listy sąsiedztwa

dla wierzchołka `a`:	`c, g`	dla wierzchołka `e`:	`d`
dla wierzchołka `b`:	`d, c, h`	dla wierzchołka `f`:	`c, g`
dla wierzchołka `c`:	`g, b, f, a`	dla wierzchołka `g`:	`c, a, f`
dla wierzchołka `d`:	`b, e, h`	dla wierzchołka `h`:	`d, b`.

Narysuj drzewa tworzone odpowiednio podczas przeszukania grafu algorytmami DFS i BFS, gdy startujemy z wierzchołka `c`. W każdym przypadku podaj kolejność, w jakiej odwiedzane będą wierzchołki grafu.

4. Graf liniowy $G^* = (V^*, E^*)$ (białe wierzchołki) grafu nieskierowanego $G = (V, E)$ (szare wierzchołki) określony jest następująco: wierzchołki $p \in V^*$ reprezentują krawędzie $e \in E$ oryginalnego grafu G, przy czym p i q połączone są krawędzią w G^* wtedy i tylko wtedy gdy odpowiadające im krawędzie e i f w G są incydentne (tj. mają wspólny wierzchołek $v \in V$), por. przykładowy rysunek. Napisz funkcję wyznaczającą macierz sąsiedztwa $B_{(m\times m)}$ grafu G^* na podstawie danej macierzy sąsiedztwa $A_{(n\times n)}$ grafu G.

Wskazówka: Trzeba ustalić jakąś numerację krawędzi e_j w G: najpierw numerujemy krawędzie wychodzące z wierzchołka 1 w kolejności wyznaczonej przez numery ich drugich końców. Dalej, w podobnej kolejności te wychodzące z 2, które nie otrzymały jeszcze numerów itd. Numery te można wstępnie zapisać wprost w elementach macierzy A: `if (A[i,j]==1) A[i,j]=A[j,i]=++k;`. Następnie w pierwotnie wyzerowanej macierzy `B` trzeba ustawić jedynki we wszystkich elementach `B[ A[i,j], A[j,l] ]`, dla których jednocześnie `A[i,j]` i `A[j,l]` są niezerowe.



5. *Ekscentrycznością* $\mathcal{E}$ wierzchołka v w nieskierowanym grafie G nazywamy długość najdłuższej wśród najkrótszych ścieżek z v do wszystkich innych wierzchołków,

$$\mathcal{E}(v) \;=\; \max_{x\in V(G)} d_{\min}(v,x)\,, \qquad d_{\min}(v,x) = \text{długość najkrótszej ścieżki z } v \text{ do } x\,.$$

Centrum $\mathcal{C}$ grafu G jest to zbiór wierzchołków G o minimalnej ekscentryczności. Wśród zastosowań można wymienić np. algorytm wyboru wsi na siedzibę gminnej przychodni zdrowia. Wybór wsi z centrum grafu połączeń drogowych jest najbardziej sprawiedliwy dla mieszkańców całej gminy, także tych z jej obrzeży.

Załóżmy, że mamy funkcję `float dmin(int u, int v)`, zwracającą długość najkrótszej ścieżki z u do v w grafie. Napisz program, który wyznaczy ekscentryczności `E[v]` wszystkich wierzchołków, a następnie centrum danego grafu.

Bonus (1 pkt): Zaproponuj jak można by dodatkowo uwzględnić liczbę mieszkańców poszczególnych wiosek w sprawiedliwym wyborze lokalizacji przychodni.

Rozwiązania

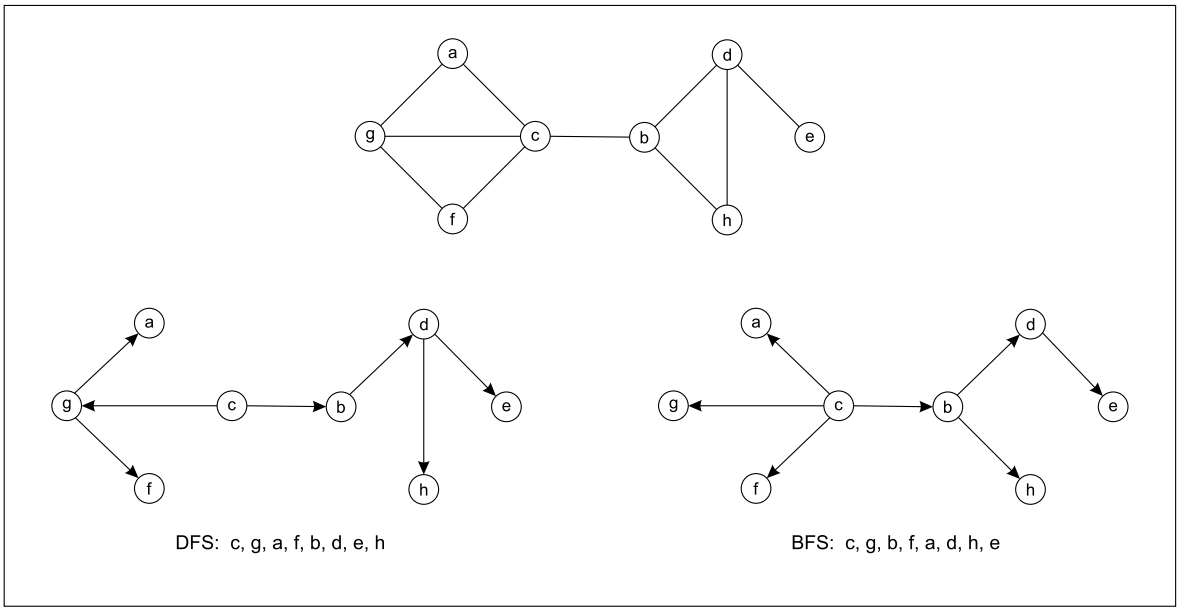
--- Zad. 1. ---

	porównania	podstawienia
tablica	$O(n^2)$	$O(n^2)$
lista	$O(n^2)$	$O(n)$

----- Zad. 2. -----

```
void merge_sort(wel *P){
   wel Q=NULL;
   if ((*P!=NULL) && (*P->nast!=NULL)) {
      podziel(P, &Q);
      merge_sort(P);
      merge_sort(&Q);
      połącz(P, &Q);
   }
}
```

----- Zad. 3. -----



----- Zad. 4. -----

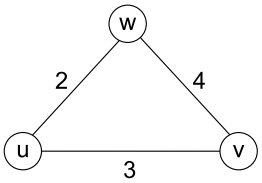
Dane - macierz sąsiedztwa A[n][n] grafu G, B[m][m]={0}, n = l. wierzchołków, m = l. krawędzi G.

```
k=0;
for (u=1; u<=n; u++)
   for (v=u+1; v<=n; v++)                //  for (v=0; ...) to błąd, bo A[u][v] przetwarzane
                                         //  są wówczas 2-krotnie z różnymi wartościami k
      if ( A[u][v]==1 ) A[u][v]=A[v][u]=++k;

for (t=1; t<=n; t++)
   for (u=1; (u<=n && u!=t); u++)
      for (v=u+1; (v<=n && v!=t); v++)
          if (A[u][t]!=0 && A[t][v]!=0)
             B[ A[u][t] ][ A[t][v] ] = B[ [A[t][v] ][ A[u][t] ] = 1;
```

```
----- Zad. 5. -----
ME = INT_MAX;
for (u=1; u<=n; u++){
   E[u] = 0;
   for (v=1; v<=n; v++){
      d = dmin(u,v);
      if (E[u] < d) E[u] = d;
   }
   if (E[u] < ME) ME = E[u];
}
printf("Centrum: ");
for (u=1; u<=n; u++)
   if (E[u]==ME) printf("%i  ", u);
printf("\n minimalna ekscentryczność %i\n", ME);
```

UWAGA: O ile w grafie nieskierowanym `dmin(u,v)`=`dmin(v,u)`, to na ogół `E[u]`≠`E[v]`.



dmin(u,v)=3, ale E[u]=3, E[v]=4

Bonus: Optymalizacja uwzględniająca liczbę mieszkańców wiosek powinna operować czymś w rodzaju "kosztów transportu", które prócz odległości `dmin` powinny być proporcjonalne do liczby dojeżdżających osób. Zamiast funkcji `dmin(u,v)` należałoby użyć np.

$$\texttt{kmin}(\texttt{u},\texttt{v}) \;=\; \texttt{dmin}(\texttt{u},\texttt{v}) \cdot \texttt{m}[\texttt{v}]$$

gdzie `m[v]` jest liczbą mieszkańców wioski `v`. Zauważmy, że funkcja `kmin` nie jest już teraz symetryczna, `kmin(u,v)` $\neq$ `kmin(v,u)`. Jeśli np. w `u` mieszka 100 osób, a w `v` 150, to przy odległości `dmin(u,v)` = 20 km mielibyśmy `kmin(u,v)` = 3000 i `kmin(v,u)` = 2000. Gdyby nie było innych wiosek, obliczenie zmodyfikowanej ekscentryczności wskazałoby — słusznie – `v` jako centrum.

Alternatywnie można też użyć "łącznych kosztów transportu" do danej miejscowości

$$\texttt{K}[\texttt{u}] = \sum_{v \neq u} \texttt{dmin}(\texttt{u},\texttt{v}) \cdot \texttt{m}[\texttt{v}]^{\gamma}$$

i wybrać jako centrum wierzchołki o minimalnej wartości `K`. Opcjonalna "korekta gamma" przy eksperymentalnie dobranym wykładniku $\gamma > 1$ mogłaby wprowadzić dodatkową preferencję dla większych miejscowości.