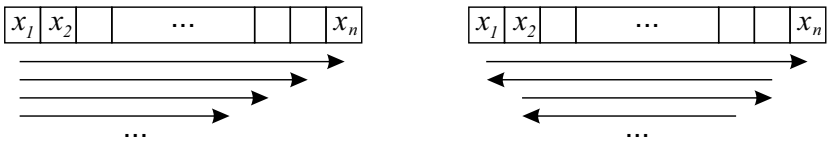
1. Zrealizować procedurę sortowania przez zamianę `void sort1(int x[], int n)`, która w `k`-tym przebiegu wyszukuje najmniejszy wśród elementów `x[k+1],...,x[n]` i zamienia go z `x[k]`.

2. Napisać procedurę sortowania przez wstawianie `void sort2(int x[], int n)`, która w `k`-tym przebiegu przesuwa element `x[k]` w lewo o odpowiednią liczbę miejsc wśród już posortowanych elementów `x[0],...,x[k-1]`. Algorytm ten przypomina popularny wśród brydżystów sposób porządkowania kart w ręku bezpośrednio po rozdaniu.

3. Zmodyfikować algorytm z zad. 2 w taki sposób, aby wyznaczenie miejsca do przestawienia elementu `x[k]` odbywało się przez połówkowe przeszukanie ciągu `x[0],...,x[k-1]`.

4. Napisać procedurę `void sort3(int x[], int n)` sortującą dane w tablicy `x` metodą bąbelkową. Następnie zaproponować ulepszenia podstawowej metody: a) wykorzystujące zapamiętaną w poprzednim przebiegu pozycję `k` ostatniej wykonanej zamiany `x[k]` $\leftrightarrow$ `x[k+1]`; b) wykorzystujące zapamiętaną w poprzednim przebiegu pozycję pierwszej zamiany `x[l]` $\leftrightarrow$ `x[l+1]`; c) zmieniającej w kolejnych przebiegach kierunek przeglądania tablicy (por. rys. poniżej); d) połączenie ulepszeń a), b) i c). Przeanalizować w jaki sposób zaproponowane ulepszenia wpływają na wykonywaną liczbę porównań i zamian elementów `x[i]` i `x[i+1]` w stosunku do rozwiązania podstawowego.



5. Metoda Shella jest modyfikacją algorytmu sortowania bąbelkowego, polegającą na wstępnym posortowaniu elementów tablicy odległych o `h` > 1, a więc `x[0], x[h], x[2h],...`, dalej `x[1], x[h+1], x[2h+1],...` itd. Kolejny przebieg działa podobnie, lecz dla mniejszej wartości `h`. W ostatnim przebiegu `h=1` i algorytm działa tak, jak zwykłe sortowanie bąbelkowe. Zysk jaki uzyskuje się w metodzie Shella w stosunku do podstawowego algorytmu polega na tym, że we wstępnych przebiegach tablica ulega częściowemu posortowaniu znacznie mniejszym nakładem pracy, gdyż duże i małe elementy przemieszczają się na koniec i odp. na początek tablicy w niewielkiej liczbie zamian wykonywanych nie między bezpośrednimi sąsiadami, lecz między elementami w znacznej odległości `h`. Doświadczalnie zbadano jakie sekwencje odległości $\mathtt{h}_k \geq \mathtt{h}_{k-1} \geq \ldots \geq \mathtt{h}_0 = 1$ dają najlepsze wyniki. Jedna z proponowanych sekwencji to liczby spełniające zależność $\mathtt{h}_j = \left(3^{j+1} - 1\right)/2$, a więc 1, 4, 13, 40 itd., przy czym największą z nich wybiera się tak, aby nie przekraczała ona jednej trzeciej rozmiaru tablicy `x`. Napisać program realizujący sortowanie metodą Shella.

6. Zrealizować algorytm Dijkstry wyszukiwania najkrótszej ścieżki między zadaną parą wierzchołków `s` i `t` w grafie skierowanym wykorzystujący funkcje obsługi kolejki priorytetowej, która przechowuje wierzchołki grafu z priorytetami równymi najmniejszym znalezionym do tej pory odległościom poszczególnych wierzchołków od źródła `s`.

7. Zrealizować algorytm Floyda-Warshalla wyszukiwania najkrótszych dróg w grafie skierowanym zadanym przez macierz odległości sąsiadów. Zmodyfikować następnie algorytm tak, aby wyznaczał on tranzytywne domknięcie grafu skierowanego.

8. Dla grafu danego przez listy sąsiedztwa zaimplementować rekurencyjny algorytm przeszukania DFS.