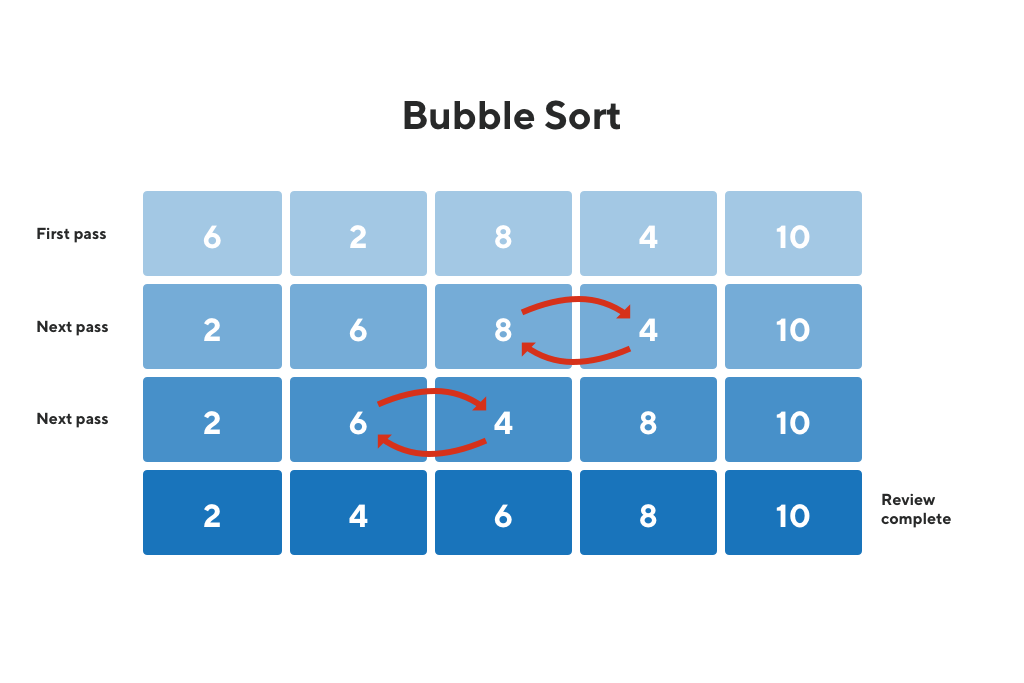
**מיון בועות רשימה מקושרת- Bubble Sort**

**מיון בועות:** אלגוריתם שבודק צמדי ערכים וממיין מבין שניהם את הערך הגדול או הקטן, הוא רץ על כל המערך/רשימה/מבנה וגורם לכך שבסוף כל מעבר הערך הגדול/הקטן ביעבע לסוף/תחילת המבנה.

**איך יתבצע המיון:** נחלק את התכנית לשלושה פונקציות: 1. בניית המבנה(הוספת מבנה ויצירת הרשימה המקושרת). 2. החלפת שני מבנים(ביעבוע) ובו תנאים רבים לכל המקרים האפשריים. 3. סידור כל הרשימה, משתמש בפוקנצייה ההחלפה ומתבצעת עד אשר כל הרשימה מסודרת לחלוטין.

**דוגמא לפני ואחרי: **

**הקוד:**

void ReplaceItem(t\_ItemNum\* item)

{

    if (item == NULL)

    {

        LogEventVal("No need to swap. No Items in list", item->num);

    }

    else if (item->next == NULL)

    {

        LogEventVal("No need to swap", item->num);

    }

    else if (item == Head && item->next == Tail)

    {

        item->next = NULL;

        item->prev = Tail;

        Tail->prev = NULL;

        Tail->next = item;

        Tail = item;

        Head = item->prev;

    }

    else if (item == Head && item->next != NULL)

    {

        item->next = item->next->next;

        item->next->prev->next = item;

        item->next->prev->prev = NULL;

        item->prev = item->next->prev;

        item->next->prev = item;

        Head = item->prev;

    }

    else if (item->next == Tail)

    {

        item->prev->next = item->next;

        item->next->prev = item->prev;

        item->next->next = item;

        item->prev = item->next;

        item->next = NULL;

        Tail = item;

    }

    else {

        // regular swap

        item->prev->next = item->next;

        item->next->prev = item->prev;

        item->next->next->prev = item;

        item->prev = item->next;

        item->next = item->next->next;

        item->prev->next = item;

    }

}

void AddItemVal(int val)

{

    t\_ItemNum\* newItem = malloc(sizeof(t\_ItemNum));

    LogEventVal("Create new Item", val);

    newItem->num = val;

    newItem->next = NULL;

    if (Head == NULL) {

        LogEventVal("List is started now ", val);

        Head = newItem;

        Tail = newItem;

        newItem->prev = NULL;

    }

    else

    {

        LogEventVal("Val is Added  ", val);

        Tail->next = newItem;

        newItem->prev = Tail;

        Tail = newItem;

    }

}

void Sort()

{

    LogEvent("Start of Sorting the list");

    t\_ItemNum\* curr = Head;

    if (!curr) {

        LogEvent("The List is empty");

        return;

    }

    char change = 1;

    while (change != 0)

    {

        change = 0;

        while (curr != NULL)

        {

            if (curr->next != NULL && curr->num > curr->next->num)

            {

                ReplaceItem(curr);

                change++;

            }

            curr = curr->next;

        }

        curr = Head;

    }

    LogEvent("End of Sorting the list");

}