שם	נושא	מטרה	סיבוכיות	Pseudo Code	סרטון	הערות
Insertion Sort	מיון	מקבל רשימה ומיין מהקטן לגדול	$O(n^2)$ השוואות	לחץ כאן	לחץ כאן	מאוד לא יעיל - הכנה לחומר
Merge	מיון	מקבל 2 רשימות ממיונות ומחזיר רשימה חדשה ממויינת	$O(\|list1\|+\|list2\|)$ השוואות	לחץ כאן	לחץ כאן	אלגוריתם שמשרת את Merge Sort
Merge Sort	מיון	מקבל רשימה ומחזיר אותה ממיונת	$O(nlog(n))$ השוואות	לחץ כאן	לחץ כאן	סגנון מיון - הפרד ומשול
Make Heap	מיון	יצירת ערימה חדשה עבור n איברים	$O(n)$ השוואות	-	-	לא ראינו מימוש
Heapify Up	תיקון ערימה	מכניס איבר לערימה מתוקנת. מלמעלה	$\leq log_2(n)$ השוואות	לחץ כאן	לחץ כאן	אלגוריתם שמשרת את Heap Sort
Heapify Down	תיקון ערימה	מכניסה איבר לערימה מתוקנה מלמטה	$\leq log_2(n)$ השוואות	לחץ כאן	לחץ כאן	אלגוריתם שמשרת את Heap Sort
Heap Sort	מיון	מקבל ערימה לא ממוינת ומחזיר ערימה ממויינת	$O(nlog(n))$ השוואות	לחץ כאן	לחץ כאן	
Select	בחירת איבר	מקבל מערך ואינדקס k ומחזיר את המיקום שלו אילו המערך היה ממוין	$O(n)$ השוואות	לחץ כאן	לחץ כאן	לפי Pivot חציון
Quick Sort	מיון	מקבל רשימה לא ממוינת ומחזיר רשימה ממויינת	$\Theta(nlog(n))$ השוואות $O(nlog(n))$ בממוצע	לחץ כאן	לחץ כאן	
Max Disjoint Intervals	אלגוריתם חמדני	בהינתן קבוצה של N מרווחים, המשימה היא למצוא את הסט	$O(nlog(n))$ השוואות	לחץ כאן	לחץ כאן	
האפמן	קידוד	מציאת קידוד אפוטימלי עבור n אותיות	$O(nlog(n))$	לחץ כאן	לחץ כאן	
Kruskal	גרפים	מקבל גרף ומוציא עץ פורש מינימלי שנמצא בתוכו	$O(\|E\|log(\|E\|)$	לחץ כאן	לחץ כאן	
Prime	גרפים	מציאת עץ פורש מינימלי בגרף שסכום המשקלים בו גם מינימלי	$O(\|E\|log(\|E\|)$	לחץ כאן	לחץ כאן	
Prime #2				לחץ כאן		אנחנו חסכוניים טיפה בגודל של הערימה ודברים קורים קצת פחות פעמים. אבל אותה סיבוכיות

שם	נושא	מטרה	סיבוכיות	Pseudo Code	סרטון	הערות
BFS	גרפים	מעבר על כל הקודקודים שאפשר להגיע להם מקודקוד u ואז בלכל קודקוד עושה משהו שצריך	$O(\|V\|+\|E\|)$	לחץ כאן	לחץ כאן	
Dijkstra	גרפים	מוצא את הדרך הקלה ביותר בין קודקוד נתון לכל שאר קודקודים בגרף	$O(\|V\|+\|E\|log(\|V\|))$	לחץ כאן	לחץ כאן	בפועל האלגוריתם מכניס לכל קודקוד את המרחק שלו מהקודקוד הנתון
DFS	גרפים	מעבר על כל הקודקודים שאפשר להגיע להם מקודקוד u ואז בלכל קודקוד עושה משהו שצריך	$O(\|E\|+\|V\|)$	לחץ כאן	לחץ כאן	דומה לBFS רק שסדר המעבר שונה
Floyd Warshall	גרפים	מציאת המסלולים הקצרים ביותר בין כל שני זוגות צמתים	$O(\|V\|^3)$	לחץ כאן	לחץ כאן	אין מעגלים שליליים - גרף מכוון וממושקל
Ford-Fulkerson with BFS	גרפים - זרימה	מיציאת זרימה אופטימלית בגרף	$O(\|E\|^2 \cdot \|V\|)$	לחץ כאן	לחץ כאן	אלגוריתם חמדן - שימוש בBFS, אחרת נקבל סיבוכיות $O(\|E\| \cdot f_{max}(e))$
רכיבי קשירות חזקה	גרפים	מציאת רכיבי קשירות חזקה	$O(\|E\|+\|V\|)$	לחץ כאן		שימוש בDFS
אוקלידס	אריתמטיקה	מציאת מחלק גדול ביותר של 2 מספרים	$O(log_2^2(n))$ סכום 2 מהספרים	לחץ כאן	לחץ כאן	
אוקלידס מורחב (רקורסיבי)	אריתמטיקה	מציאת מחלק גדול ביותר של 2 מספרים	$O(log(n))$ סכום 2 מהספרים			
Inverse	אריתמטיקה	מציאת הופכי של מספר	$O(log(n))$	לחץ כאן		
מציאת $a^b mod m$	אריתמטיקה	בהינתן $a,b,m \leq n \in \mathbb{Z}$ נחשב את $a^b mod m$	$O(log^3(n))$	לחץ כאן		

אלגוריתמים נוספים מהתרגולים ומההרצאות

מטרה	נושא	סיבוכיות	אלגוריתם עזר	הערות
מיזוג 2 רשימות ממוינות עם גדלים n וk ויוצרת רשימה חדשה	מיון	$O(klog(n))$ השוואות	חיפוש בינארי	הגדלים לא חייבים להיות שווים
מיזוג k רשימות ממוינות לרשימה ממוינת אחת (סה"כ n איברים)	מיון	$O(nlog(k))$ השוואות	Merge	
אלגוריתם שמקבל רשימה ממוינת A באורך n. ומחלק אותה לk קבוצות שוות כך שלכל i כל איבר בA_i קטן מכל איבר שA_{i+1}	מיון	$O(nlog(k))$ השוואות	Merge ? Select	
מציאת איבר מינימלי ומקסימלי ברשימה באורך n	מציאת איבר	$\leq \frac{3}{2} \cdot n - 2$ השוואות		
מוט באורך n ווקטור P של מחירים של מוט באורכים שונים. מרצה לחתוך את המוט כך שנקבל כמה שיותר כסף	תכנון דינמי	$O(n^2)$	שימוש במערכים לשמירת המידע	
מציאת מספר הפעולות כפל המינימלי שהכרחי לחישוב למכפלת כל המטריצות (n)	תכנון דינמי	$O(n^3)$		גדלי המטריצה שונים ?
עבור צמתים s,t בגרף ממושקל, מציאת כל הצמתים ששייכים למסלול הקל ביותר ביניהם	גרפים	$O(\|V\| + \|E\| log(\|V\|))$	Dijkstra	
מציאת מs לכל v בגרף אורך הקצר ביותר של מסלול באורך זוגי ביניהם	גרפים	$O(\|V\| + \|E\|)$?	BFS הכפלה של הגרף	
מציאת זיווג מקסימלי בגרף דו-צדדי	זרימה	$O(\|V\| + \|E\|^2)$	Ford-Fulkerson	זיווג - היא קבוצת קשתות זרות
שיבוץ משמרות לתאריכים עם מגבלות ספציפיות	זרימה	$O(n + \|A\| + \|B\| + n \cdot m + n \cdot \|A\|)$	Ford-Fulkerson	תרגול 12

אלגוריתמים מגיליונות (ככל הנראה יש עוד, בחרתי מה שלא היה ספציפי מידי - ללא טענות):

מטרה	נושא	סיבוכיות	אלגוריתם עזר	גיליון
מיון מערך לא ממוין בגודל n של מספרים בטווח $1,2,....,2n$	מיון	$O(n)$ השוואות		גיליון 1
מציאת חציון של 2 רשימות ממוינות (n,m)	מיון - חישוב חציון	$O(log(n))$ השוואות		גיליון 2
מציאת האיבר השני הכי גדול ברשימה לא ממוינת	מיון - חישוב מקסימלי שני	$\leq n + \lceil log_2(n) \rceil - 2$ השוואות		גיליון 2
מציאת חלוקת של V ל10 חלקים זרים כך שיהיו רחוקים זה מזה	גרפים	$O(\|E\| log(\|E\|))$ עם דלתא גדול ככל האפשר		גיליון 3
כמה רכיבי קשירות יש בגרף	גרפים	$O(\|V\| + \|E\|)$	BFS	גיליון 4
האם יש בגרף מעגל				
האם יש גרך במעגל באורך אי-זוג או לא				
כמה מסלולים יש בגרף בין 2 קודקודים נתונים באורך הקצר ביותר				
האם צלע נתונה היא שייכת לעץ פורש מינימלי				
שיפור FF. כדי לקבל את גם את המסילה הקצרה ביותר מ-u ל-v (לא רק את המרחק) ולהוסיף אלגוריתם שבודק אם המטריצות שחזרו נכונות	גרפים	$O(\|V\|^2 + \|E\| \cdot \|V\|)$	Floyd-Warshall	גיליון 4
תרגילים נוספים של תכנון דינמי של סדרה עם תתי סדרות עולות				גיליון 4
חילוק את קבוצת הקודקודים לשני חלקים לא ריקים, כך שמספר הצלעות עם קודקוד בחלק אחד וקדקוד בחלק השני הוא קטן ככל האפשר (גרף לא מכוון)	גרפים	$O(\|V\|^4)$		גיליון 5
מהם כל הקודקודים שאפשר להגיע אליהם במסלול מכוון מכל קודקוד אחר בגרף	גרפים	$O(\|V\| + \|E\|)$	DFS מיון טופולוגי רכיבי קשירות חזקה ותכונותיהם	גיליון 5
נתון קודקוד v, מכמה קודקודים אפשר להגיע ל-v במסלול מכוון לא פשוט				
ניתן למצוא את המעגל המכוון הראשון שנוצא מצלעות שהDFS כבר בדק אותו				
בהנחה שאין בגרף מעגלים, ונתונים 2 קודקודים, כמה מסלולים יש ביניהם				
ניתן לבדוק אם לכל שני קודקודים u,v אם יש מסלול u->v או u<-v				

מסקנות וטענות מהגיליונות:

1. גיליון 1:
    1. כככ
2. גיליון 2:
    1. כככ
3. גיליון 3:
    1. כככ
4. גיליון 4:
    1. כככ
5. גיליון 5:
    1. כככ

Insertion Sort .1

```
def Insertion_Sort(List list):
    for i=2 in list.lenght: #every time we look in first i nodes in list
        k=list[i]
        j=i-1
        while j>=1 and  k< list[j]: #every time we move the new node to the right place in the underlist we look at
            list[j+1]=a[j]
            j=j-1
        list[j+1]=k
    return list

```

Merge .2

```
def merge (List a,List b):
    mergedlist = createListInSize(a.size+b.size)#make empty list size a+b
    i,j=1
    for m=1 to a.size+b.size:
        if j<=a.size and (i>b.size or a[j]<b[i]):
            mergedlist[m] = a[j]
            j++
        else:
            mergedlist[m]=b[i]
            i++
    return mergelist
```

Merge Sort .3

```
def merge_sort(List a):
    n = a.size
    if n<=1: return a
    b = merge_sort(a[1],...,a[n/2])
    c = merge_sort(a[n/2],....,a[n])
    return merge (b,c)
```

Heapify Down .4

```
def heapify_down (Heap h,Node v):
    while left(v)<=n:
        if right(v)<= h.size and (h[right(v)]>h[left(v)]):
            u=right(v)
        else:
            if h(u)>h(v):
                swap(h(u),h(v))
                v=u
            else return
```

Heapify Up .5

```
def heapify_up (Heap h,Node v):
    while v>1 and v.parent>v:
        swap (h[v],h[v.parent]
        v=parent(v) #update the pointer who need to heap up

```

Heap Sort .6

```
def Heap_sort (Heap h):
    start_heap(h)
    for v in (n,n-1,...,2):
        swap (h[v],h[1])
        heapify_down(h(1,...,v-1),h[1])
```

Select .7

```
def Select(Array A,k):
    n=A.size
    if n <=5:m=A[1]
    else:
        F=(A[1,..,5],A[6,...,10],...,A[...,floor(n\5)*5])
        M=(Select(F[1],3),Select(F[2],3),...,Select(F[floor(n\5)],3))
        m=Select (M,floor(n\10))
    L={A[i]|A[i]<m}
    R={A[i]|A[i]>m}
    if |L|=k-1:return m
    if |L|>k-1:return Select(L,k)
    if |L|<k-1:return Select(R,k-|L|-1)
```

Quick Sort .8

נתבונן באלגוריתם הבא: בהינתן רשימה A באורך n:

1. **בוחרים** $a_k \in A$ $(pivot)$
2. **נגדיר רשימות** $B = \{a_i \in A | a_i \leq a_k\}$ **ו־** $C = \{a_i \in A | a_i > a_k\}$ **(אם** c **ריקה - נבחר** a_k**אחר)**
3. **נמיין את** B, C **ריקורסיבית ונחזיר** (B, C)

Max Disjoint Intervals .9

```
def Max_Disjoint_Itervals((a[1],b[1]),...,(a[n],b[n])):
    sort by b[i]
    I={i}
    i=1
    for j in (2,...,n):
        if a[j] >= b[i]:
            I=I united with {j}
            i=j
    return I
```

10. האפמן

אותיות C, $p(c)$ לכל אות $c \in C$

1. הכניסו את העלים c עם $p(c)$ לערימת מינימום ממוינת ע"י $p(c)$
2. בצע $|c| - 1$ פעמים:
   - (א) הוציא את x, y מראש הערימה
   - (ב) הכנס לערימה את z שהוא עץ עם 2 בנים x, y והסתברות של z הוא $p(z) = p(x) + p(z)$
3. החזר האיבר שבראש הערימה

11. Kruskal

```python
def kruskal(n,E):
    T=[] #empty vector
    E=sorted(E,key=lambda e:e[2]) #sorted by w
    C=[[v] for v in range(n)]
    for u,v,w in E:
        if C[u] != C[v]:
            T.append((u,v))# and (u,v) to T
            if len(C[u])>len(C[v]): u,v=u,v #switch names
            C[v].extend(C[u])# C[v]<--C[v]UC[u] (union by value)
            for x in C[u]:C[x]=C[v].#change to "pointer"
    return T
```

12. Prime

```python
def prim(graph G, vertex v): #v is random vertex to start with
    T = graph(V={v},E=empty)
    H = start_heap({(u,v): u in G.neighbors(v)}) #order by weight (u,v)
    for k in {2,3,.....,|G.V|}: #G.V num of vertexes
        (u,v) = H.pop() #get first edge
        T.add_vertex(u)
        T.add_edge((u,v))
        for x in G.neighbors(u):
            if x in T: H.remove((u,x)) # x already in T, no circle needed
                and we already add the shortest edge with x
            else : H.add ((x,u)) #x not in T
    return T
```

13. Prime #2

```python
def prim(graph G, vertex v): #v is random vertex to start with
    T = graph(V={v},E=empty)
    H = start_heap({(u,v): u in G.neighbors(v)}) #order by weight (u,v)
    for k in {2,3,.....,|G.V|}: #G.V num of vertexes
        (u,v) = H.pop() #get first edge
        T.add_vertex(u)
        T.add_edge((u,v))
        for x in G.neighbors(u):
            if x not in T:
                (x,y) = H.find((x,*)) #* := any vertex
                if (x,y).weight > (x,u).weight:
                    H.replace((x,y),(x,u)) # switch to (x,u)
    return T
```

14. BFS (נניח ורוצים למצוא רכיבי קשירות)

```
def BFS (graph G,vertex u):
    S=[]
    Q=[u] # queue - first in first out, insert from right, out from left
    for v in G.V:
        v.visited = false
    u.visited = true
    while Q not empty:
        v=Q.pop_from_left()
        for x in G.neighbors (v):
            if not x.visited:
                Q.add_from_right(x)
                x.visited = true
        S.append(v)
    return S
```

15. Dijkstra

```
def dijkstra(G,w,u):
    for x in G.V:
        x.d=infinity
        x.p=null
    u.d=0
    H=start_heap(G.V,order in minimum heap by x.d)
    while H not empty:
        v=H.pop() #add to S
        for x in G.neighbors(v):
            if x in H and (v.d+w(v,x)<x.d):
                x.d=v.d+w(v,x)
                x.p=v
                H.heapify(x) #we need to save x place in the heap to make
                    this
    return G # now include distance and routes
```

16. DFS

```
def DFS(G,u):
    u.visited=True
    for v in G.out_neighbors(u):
        if not v.visited:
            DFS(G,v)
```

```
def DFS_main(G,u):
    T=[] #global varibale
    time = 0 #time is global variable
    for v in G.V:
        v.color = WHITE
    for u in G.V:
        if u.color==WHITE:
            DFS_visit(G,u)
    return T

def DFS_visit(G,u):
    u.color=GRAY
    time +=1
    a.d=time
    for v in G.outneighbors(u):
        if v.color==WHITE:
            DFS_visit(G,v)
            T.append((u,v))
    u.color = BLACK
    time +=1
    a.f = time
```

17. Floyd-Warshall

```
def APSP_Floyd_Warshall(G,W):
    def n times A(n,n) #make n+1 matrix n x n
    if (i=j):
        A[0](i,j)=0
    else:
        A[0](i,j)=w(i,j)
    for k=1 to n:
        for i,j:
            A[k](i,j)=min{A[k-1](i,j),A[k-1](i,k)+A[k-1](k,j)}
    return A[n]
```

18. רכיבי קשירות חזקה

10.3.2 אלגוריתם למציאת רכיבי קשירות חזקה

הגדרה: יהי $G=(V,E)$ גרף־ מכוון. נגדיר $u \sim v$ אם קיים מסלול מu ל־v, ולהפך.
היחס $u \sim v$ הוא **יחס שקילות**. מחלקות השקילות נקראות רכיבי קשירות חזקה של G

האלגוריתם: בשלבים הבאים:

(1) - הפעל DFS על כל הגרף G ונשמור את זמני הסיום $f(v)$ של כל קודקוד

(2) - הפעילו DFS על הגרף ההפוך כאשר בחירת הקודקודים היא לפי f בסדר יורד

כלומר, כל עוד נותרו קודקודים בגרף ← יהי v_i קודקוד עם $f(v_i)$ מקסימלי ($i=1,2,\ldots$) ← הפעל $\text{dfs\_visit}(v_i)$
עם צלעות מכוונות הפוך.

הסירו את הקדוקודים שעברנו עליהם והגדירו אותם בתור רכיב קשירות חזקה V_i

19. Ford-Fulkerson With BFS

<u>האלגוריתם:</u> דומה למקודם

1. נתחיל מזרימת האפס $f(e)=0$ לכל e
2. נחפש מסלול $s-t$ עם עודף חיובי - בונים את גרף המסלולים G_f מפעילים BFS (קצר ביותר)
3. נגידר מחדש f יותר טובה
$$f'(e)=\begin{cases} f(e) & e\notin P \\ f(e)+excess(f,p) & e \text{ with the flow} \\ f(e)-excess(f,p) & e \text{ against the flow} \end{cases}$$
4. אם אין כזה מסלול נעצור, ולפי המשפט f זרימה מקסימלית

20. אוקלידס ואוקלידס המורחב

```
def gcd(a,b):
    while a!=0:
        a,b=b%|a|,a #that line do r=b%a , b=a,a=r
    return b
```

```
def extended_gcd(a,b):
    if a==0:
        return (b,0,1) #return (b,x,y)
    (q,r)=divmod(b,a)#insert the result to q and remainder part to r
    (d,x,y)=extended_gcd(r,a)
    return (d,y-q*x,x)
```

21. Inverse

```
def inverse(a,n):
    d,x,y=gcd(a,n)
    if d not equal 1: return None
    return x
```

22. מציאת $a^b mod m$

```
Answer:
    Def c=0,d=1
    Find b=<b[k],b[k-1],...,b[1] #binary form of b
    for i=k to 0:
        c=2c
        d=d*d mod (m)
        if b[i]=1:
            d=d*a mod(n)
            c=c+1
    return d
```