```
הגדרות לאלגוריתמים של עץ skeleton:
                                                                                                                                                                                                                          נכתב ע"י צבי מינץ ויואב ג ע"ג בי מינץ ויואב ג ג
גַּגָּבָּם מרצה :פרופ' דנה שפירא
                                                                                                                                                                                                                    נכתב ע"י צבי מינץ ויואב גרוס 🖫
                                 1. אם צמד תווים מופיע פעמיים בצד ימיו הוא יוחלף
                                                                       base(i) = 2(base(i-1) + n_{i-1}) Golomb decode(b)
                                                                                                                                                 Golomb_encode(x,b)
 מצא את הזוג השכיח ביותר והחלף אותו
                                                                              seq(i) = seq(i-1) + n_{i-1} \quad q \leftarrow Unary\_decode()-1;
                  במשתנה חדש.
                               2. אם משתנה מופיע פעם אחת בצד ימין – נמחק אותו.
                                                                                                                                                       q \leftarrow (x-1) \text{ div } b;
                  (בהינתן מילון עם מילות קוד באורך קבוע, יש להתאים מחרוזות) Tunstall
                                                                              diff(i) = base(i) - seq(i) r\leftarrowMinimal binary_decode(b); r\leftarrowx-q·b;
                                                                                                                                                                                        (בסוף רוצים קוד K(\mathcal{C}) \leq 1 מינמלי E(\mathcal{C}, P) מינמלי מבוא:
                                                         .Σ הכנס למילון את
                                                                                                                                                                                         E(C,P) = \sum_{i=1}^{n} p_i \cdot |c_i| אורך מילת הקוד הממוצעת:
                                                                                                                                                      I = Unary encode(q+1);
                                                                                                               return r+a·b:
                                                                         :Decoding Algorithm
                                             כל עוד נותר מקום לΣ תווים נוספים:
                                                                                                                                                                                                                     I(s_i) = -\log_2 p_i אינפורמציה:
                                                                                                                                                     n = Minimal_binary_encode(r,b);    return l·n
                                                                        1. tree_pointer<-root
         . למילון, הכנס את את למילון למילון ההסתברות המקסי' למילון למילון למילון למילון למילון למילון למילון למילון למילון 1
                                                                                                                                                                                          H(P) = -\sum_{i=1}^{n} p_i \log_2 p_i אנטרופיה (חסם תחתון):
                                                                                                                             . כלשהו k להיות 2^K עבור b נבחר את :Rice code
                                                                                                        Skeleton
                                             2. אם \Sigma \neq d מחק את d מהמילון.
                                                                                                                                                                                                                         \forall C: H(P) \leq E(C, P) כלומר
                                                                                                                    חלק ראשון: קוד אונרי של (right shift k bits)) חלק ראשון
                                                           קוד פיבונאצ'י 1
                                                                                                                                                                                         C' אזי קיים קוד K(C) = \sum_{i=1}^{n} 2^{-|c_i|} \le 1 אזי קיים קוד
                                                                         4. while i<length_of_string
                                                                                                                       X-1 (לא תמיד הבי יעיל) Shannon-Fano Algorithm:
                   כתוב את המספר בבסיס פיבונאצ'י, הוסף 1 משמאל, הפוך את המספר.
                                                                                                                                                                                          יסר ש- C'-ו |C| = |C'|, E(C, P) = E(C', P) פרפיקסי
                                                                                  if string[i]=0
 k+1 הן באורך F_{k+1} \leq j < F_{k+2} הקוד בטווח מיידי. כל מילות מיידי. כל מילות הקוד בטווח
                                                                                                                                                 מיין את ההסתברויות בסדר יורד.
                                                                                       tree pointer<-left(tree pointer)
                                                                                                                                                                                           אם K(\mathcal{C}) > 1 אזי לא קיים קוד פרפיקסי עם האורכים
                                                           2 קוד פיבונאצ'י 4.2.
                                                                                  else tree_pointer<-right(tree_poi
                                                                                                                                                                                                     מלא פוצר עץ מלא K(\mathcal{C}) = 1. קוד שלם: K(\mathcal{C}) = 1
                                                                                                                                  כל עוד קבוצת ההסתברות מכילה יותר מתו 1:
                                             Fib2(n+1) עבור, Fib2(1) = 1 4.3.
                                                                                  if value(tree_pointer)>0
                                                                                                                    -נחלק את הקבוצה ל2 חלקים כך שסכום ההסתברויות בכל
                                                                                                                                                                                         קוד חסר רישות ⇒ קוד UD (ניתן לפענוח בצורה יחידה)
                                                                                       codeword<-string [start ...
                       משמאל. 10 כתוב את n בבסיס פיבונאצ'י, הפוך את המספר, הוסף 10 משמאל.
                                                                                                                                                                                      קוד מיידי ⇔ קוד חסר רישות, קוד UD ≠קוד חסר רישות
                                                                                                                     חלק פחות או יותר זהה. -קבוצה אחת תקבל 1, והשנייה 0.
                                                       (start+value(tree pointer)-1)] דחיסה מילונית
                                                                                                                                             קוד הפמן: אלגוריתם לבניית עץ קנוני
                                                                                                                                                                                                                      מבחו לזיהוי קוד יחודי: (אם"ם)
       את לחפש אז לחפש את \sigma נמצא אז לחפש את – אפשר ע"י גרידי (אם 4.3.2.
                                                                                       output<-table[I(codeword)-
                                                                                                                        (l_1, ..., l_n) אחרי הרצת הפמן קיבלנו את כל האורכים
                                                                                                                                                                                                                                                                  \sigma היא רישא של xס ולכן סיפא מתנדנדת היא xאם x היא מילת קוד מקורית – הקוד לא ייחודי
              וכו׳ ואז להתחיל מחדש),אפשר למצוא אופטימלי \sigma \cdot \sigma_2
                                                                                                                                                                                   • Examine all pairs of codewords:
                                                                        diff[value(tree_pointer)]]
                                                                                                                        1.maxlength ←האורך של המילה הארוכה ביותר
 ע"י רדוקציה לגרף. דחיסה אדפטיבית – המילון נבנה בזמן הקידוד
                                                                                       tree pointer←root
                                                                                                                                                                                      Construct a list of all codewords.
    ומכיל שלשות Sliding Window, Look Ahead Buffer עם – LZ77
                                                                                       start<-start+value(tree_pointer) 2. for i = 1 to maxlength do \{num[i] = 0\}
                                                                                                                                                                                  2. If there exist a codeword, a, which is a prefix of
                  (0,0,\sigma) כאשר לקדד תו בודד זה (off,len,symbol)
                                                                                       i<-start
                                                                                                                        3. for i = 1 to n do \{num[l_i] + +\}
                                                                                                                                                                                       another codeword, b, add the dangling suffix to the
                                         : LZ77 אלגוריתם קידוד
                                                                                 else i++
                                                                                                                                                                                       list (if it is not there already), until:
                                                                                                                        4.firstcode[maxlength] \leftarrow 0
                                                                         1. tree_pointer<-root
         1. p←1 // The next character to be coded
                                                                                                                                                                                          You get a dangling suffix that is an original codeword \rightarrow the
                                                                                                                        5. for i = maxlength - 1 downto 1 do
                                                                        2. i <- start<-1
         2. while there is text remaining to be coded
                                                                        3. while i<length of string
                                                                                                                        5.1.firstcode[i] \leftarrow (firstcode[i+1] +
                                                                                                                                                                                          There are no more unique dangling suffixes \rightarrow the code is
         2.1. search for the longest match for S[p..] in S[p-
                                                                               if string[i]=0 tree_pointer<-
                                                                                                                                                              num[i+1])/2
         W...p-1] if the match occurs at position m with
                                                                        left(tree_pointer)
                                                                                                                        6. for i = 1 to maxlength do
                                                                                                                                                                                            מודלים: (ללמוד ולעשות הנחות על מבנה הטקסט)
         length I
                                                                               else tree_pointer<-right(tree_pointer)
                                                                                                                                                                                                 Lossy - Lossless – יש 2 סוגים של דחיסות
ex.Video ex.Text – יש 2
                                                                                                                        6.1. nextcode[i] \leftarrow firstcode[i]
         2.2 Output the triple (p-m, I,S[p+l])
                                                                               if value(tree_pointer)>0
                                                                                                                        7. for i = 1 to n do
                                                                        3.3.1.
                                                                                  len<-value(tree_pointer)
                           length > offset הצבה עצמית – כאשר
                                                                                                                                                                                (prelude אין |\Sigma| = 256(Ascii) כאשר p_i = \frac{1}{|\Sigma|} (אין
                           (AVL) שיפור, משתמש בעץ בינארי מאוזן – LZSS
                                                                        3.3.2.
                                                                                  codeword<-string[start...(start+len-1)]
                                                                                                                        7.1.codeword[i] \leftarrow nextcode[l_i]
                           ומחזיק ביט + (off,len) או ביט + אסקי (תו בודד)
                                                                                  if flag(tree_pointer)=1 and
                                                                                                                        7.2.symbol[l_i, nextcode[l_i] - firstcode[l_i]] \leftarrow i
 w ← first char of input
                                                                                                                                                                                     |\Sigma|= כך p_i=rac{1}{|\Sigma|} בטקסט p_i=rac{1}{|\Sigma|}
                           Index | Phrase שימוש בעץ, trie שימוש בעץ – LZW
                                                                        2I(codeword)>=base(len+1)
                                                                                                                        7.3.nextcode[l_i] + +
                      ומאותחל עם תווים בודדים, בנוסף מחזיק טבלה w | k
 repeat
                                                                                      codeword<-string[start...(start+len)
                                                                                                                                                                                E(C,P)=H(P)+rac{8\cdot|\Sigma|+8}{|Text\ Size|} ולכן ולכן אחבסס על מודל אסקייE(C,P)=H(P)ולכן
                                                     אלגוריתם לקידוד:
                                                                        3.3.3.2.
 k ← next char
                                                                                                                                 אלגוריתם לפענוח של הפמן קנוני ע"י הטבלאות:
                                                                                 output<-table[I(codeword)-diff[len]]
   if (EOF) output code(w)
                                                                                                                                                                                  סמי סטטי עם הסתברויות עצמיות: p_i = rac{v_i}{m} כאשר מספר
                                                                                                                        1. v \leftarrow nextInputBit()// קבלת הביט הראשון
                                                                                 tree_pointer<-root
   else
                                                                                 i<-start<-start+len
                                                                                                                        2.i = 1
                                                                                                                                                                               כמה עולה לעביר .m בטקסט בגודל ב-טקסט בs_i בטקסט באודל
     if ((w \cdot k) \in Dictionary) then w \leftarrow w \cdot k
                                                                               else i++
                                                                                                                        3. while v < firstcode[i] do
                                                                                                                                                                                               E(C,P)=H(P)+rac{8\cdot|\Sigma|\cdot|P|+8}{|Text\ Size|}הסתברות ולכן
              output code(w)
                                                                                               \mathsf{pw} = \mathsf{bound}\left(s_i\right) \leftarrow \sum_{i=1}^{n} p_i
                                                                                                                        3.1. v \leftarrow 2v + nextInputBit()
              Dictionary ← w k
                                                                                                                                                                                                     קודים: (רוצים למצוא קודים בסדר גודל לוגרתמי)
                                                                                               igh \quad bound(s_i) \leftarrow \hat{\sum}_i
                                                                       low \leftarrow 0.0
                                                                                                                        אם יצאנו מהלולאה אזי v מכילה מילת קוד תקינה//
                                                                                                                                                                                                        קודים מסדר ראשון: ישנה תלות בין כל זוג תווים
                                                                                                                        4.return\ symbol[i, v-firstcode[i]]
                                                                                                                                                                                                                    Unarv(X) = 1^{|X-1|} \cdot 0: Unary Code
אלגוריתם פענוח ל LZW: אתחל מילון עם תוים בודדים
                                                                                                                                         אלגוריתם לעדכון עץ הפמן דינמי:
OLD = first input code
                                                                                                                                                                                               [\log_2 n] כל סימן מייצג מילת קוד באורך: Binary Code:
output translation of OLD
                                                                           high ← low + high bound(symbol)*range
                                                                                                                                                                                        יהיו 2^{\lceil \log_2 n \rceil} - n אם יש: Minimal Binary Code
while not end of input stream{
                                                                                 ← low + low bound(symbol)*range
                                                                                                                            replace q by a parent 0-node with two 0-node
                                                                                                                                                                                         \lfloor \log_2 n \rfloor באורך באורך אורך ביטים ושאר באורך ביטים ושאר באורך באורך ו
 NEW = next input code
                                                                        Output any value in [low, high]
 if NEW is not in the string table
                                                                                                                       if q is a sibling of a 0-node
                                                                                                                                                                                     של מספר הביטים Unary גמה): החלק הראשון זה C_{\mathbf{v}} : Elias
      S = translation of OLD
                                                                                 אלגוריתם לפענוח קוד אריתמטי בטווח [0,1]:
                                                                                                                                                                                           ללא '1' הראשון X ליצוג I והחלק השני זה קוד הבינארי של
      S = S \cdot C
                                                                       encoded ← Get (encoded number)
                                                                                                                            increment q's weight by 1;
                                                                                                                                                                                        X של מספר הביטים ליצוג \mathcal{C}_{\nu} של מספר הביטים ליצוג החלק הראשון זה \mathcal{C}_{\delta}
                                                                                                                            q = parent of q
                                                                        do{ Find symbol whose range contains encoded
                                                                                                                       while q is not root
      S = translation of NEW
                                                                        Output the symbol
                                                                                                                            interchange q with the highest numbered node
                                                                                                                                                                                                     והחלק השני זה קוד הבינארי של X ללא '1' הראשון
                                                                                                                       of the same weight;
                                                                        range \leftarrow high(symbol) – low(symbol)
  C = first character of S
                                                                                                                            increment q's weight by 1;
                                                                        encoded \leftarrow (encoded – low(symbol))/range
  Translation(OLD) · C to the string table
                                                                                                                           g = parent of g
                                                                                                                                                                                        C_{v}: 1 + 2 · \lfloor \log_2 x \rfloor bits \lfloor C_{\delta}: 1 + 2\lfloor \log_2 \log_2 2x \rfloor + \lfloor \log_2 x \rfloor bits
                                                                                                                       increment q's weight by 1
  OLD = NEW }
```