למידה ותכנון במערכות דינאמיות
ש.ב. מס' 1

שאלה 1

.a

$$\Psi_1(2) = 1 \quad (X = 2)$$
$$\Psi_2(4) = 3 \quad (X = 1 + 3 = 3 + 1 = 2 + 2)$$
$$\Psi_3(2) = 0 \quad (no\ possible\ 3 - number\ combination)$$
$$\Psi_N(N) = 1 \quad (X = 1 + 1 + 1 + \cdots + 1)$$
$$\Psi_1(X) = 1 \quad (X = X\ is\ the\ only\ combination)$$

.b תחילה, אציג את הבעיה באופן המאפשר פתרון קומבינטורי, והפתרון יוביל לנוסחה רקורסיבית באופן מידי.

נניח שברשותנו מערך בגודל X, אותו נרצה לחלק ל-N תתי מערכים. לשם כך, יש לבחור $N-1$ אינדקסים מתוך הקבוצה $\{1, \dots, X-1\}$, שיסמנו תחילת קטע חלוקה חדש. ברור, על כן, שיש $\binom{X-1}{N-1}$ אפשרויות שונות לחלוקה.

כעת, נגיד שכל תת קטע במערך מייצג מספר, שגודלו כמספר האיברים בתת המערך. ברור שחלוקת המערך ל-N קטעים שקולה לחלוקת המספר X ל-N מספרים המסתכמים אליו, וכן ברור שחלוקת המערך רגישה לסדר החלוקה – חלוקת מערך בגודל 3 לקטע באורך 1 שלאחריו קטע באורך 2 תהיה שונה מחלוקה לקטע באורך 2 שלאחריו קטע באורך 1.

עם כן, ניתן לומר כי

$$\Psi_N(X) = \begin{cases} \binom{X-1}{N-1} & 1 \le N \le X \\ 0 & N < 1, N > X \end{cases}$$

מנוסחה זו ניתן להגדיר את הקשר הרקורסיבי הבא:

- $\Psi_1(X) = 1$
- $\Psi_{N+1}(X) = \binom{X-1}{N} = \frac{(X-1)!}{(X-1-N)!N!} = \frac{(X-1)!}{(X-N)!(N-1)!}\frac{X-N}{N} = \binom{X-1}{N-1} \cdot \frac{X-N}{N} = \Psi_N(X) \cdot \frac{X-N}{N}$

הערה: קשר רקורסיבי אחר, שיהיה פחות יעיל חישובית, אבל יותר אינטואיטיבי, מתבסס על:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$
$$\Rightarrow \Psi_N(X) = \Psi_{N-1}(X-1) + \Psi_N(X-1)$$

המשמעות: כדי לראות כמה סכומים אפשריים קיימים, נפריד לצירופים שהאיבר הראשון בסכום הוא 1 (האיבר הראשון למעלה), ולצירופים שהאיבר הראשון אינו 1: אפקטיבית, ניתן לחסר מ-X ומהמספר הראשון בסכימה 1, ונשאר עם הבעיה המקורית, ולכן זהו האיבר השני בסכום לעיל.

סיבוכיות הריצה תהיה מסדר גודל של $O(NX)$, שזה נורא איטי ביחס לחישוב הרקורסיבי לפי העצרת...

.c

.1 קוד המטלב:

```
function count = DP_counting(X,N)

if(N<0 || N>X) % out of range
    count = 0;
    return
end

if(N>X/2) % Reduce number of iteration
```

```
    N = X-N+1; %(X-1) choose (N-1) equals (X-1) Choose (X-N) -
> same as N<-(X-N+1)
end

count = 1; % Initialization
for i = 1:N-1 % recursive formula
    count = count*(X-i)/i;
end;
```

סיבוכיות הזיכרון: $O(1)$ (צריך לשמור רק את הדגימה הקודמת.

סיבוכיות זמן הריצה: $O(\min(N, X-N))$, לפי מספר האיטרציות בלולאה, כשנעזרנו בטריק

$$\binom{X-1}{N-1} = \binom{X-1}{X-N} = \binom{X-1}{(X-N+1)-1}$$

כדי להפחית זמן ריצה עבור ערכי N שקרובים ל-X.

.2

$$\Psi_{12}(800) = 1.980760285431223 \cdot 10^{24}$$

d. כעת, לכל מספר טבעי i משויך מחיר c_i. נרצה למצוא $\{x_i\}_{i=1}^{N}$, המקיימים $\sum_{i=1}^{N} x_i = X$, ושהמחיר $\sum_{i=1}^{N} c_{x_i}$ יהיה מינימאלי.

1. נגדיר את אוסף המצבים S ע"י $s_k = \sum_{i=1}^{k-1} x_i$, כלומר כל מצב ייצג את סכום המספרים עד כה. ברור כי הערכים האפשריים של s_k הינם $\{1, \dots, X\}$. בזמן הראשון ניתן להגיד כי $s_1 \in \{1, \dots, X\}$.
נגדיר את אוסף הפעולות בתור המספר אותו מוסיפים לסכום, כלומר $a_k = x_{k+1}$, ולכן $a_k \in \{1, \dots, X\}$.
פונקציית המחיר המצטברת הינה (בהגדרת הבעיה שלנו):

$$C_N(X) = \sum_{i=1}^{N-1} c(s_i, a_i) + c(s_1) = c_{s_1} + \sum_{i=1}^{N-1} c_{a_i}$$

כאשר $\sum_{i=1}^{N} a_i = X$.

2. ניעזר בנוסחה הרקורסיבית:

$$C_N(X) = \max_{i \in \{1, \dots, X-1\}} \{c_i + C_{N-1}(X-i)\}$$

שנובעת מכך שאם הסכום של N-1 איברים אינו אופטימאלי, בוודאות ניתן לשפר את הסכום של N האיברים. לכן, מספיק להסתמך על הסכום של $N-1$ איברים, ולהוסיף עוד איבר בצורה אופטימאלית.

כעת, לכל $2 \le k \le N$, נמצא את המחיר האופטימאלי לכל מצב אפשרי ב-s_k, ואת הפעולה שהובילה אליו, $a_k - 1$. החישוב לכל מצב דורש X פעולות, וישנם X מצבים, ולכן, לכל נקודת זמן במכונת מצבים יש לבצע X^2 פעולות. מכיוון שישנן N נקודות זמן, סיבוכיות הריצה הינה $O(X^2N)$. לכל מצב במכונה צריך לזכור מספר אחד לצורך בניית הסכום האופטימאלי – סיבוכיות זיכרון של $O(XN)$ (אם רק רוצים למצוא את המחיר האופטימאלי, מספיקה סיבוכיות זיכרון של $O(X)$).

הערה: ניתן לייעל מעט את מכונת המצבים שהוצעה, למשל, בזמן k, לא ייתכן מספר הגדול מ:
$X-(N-k+1)$
ולמספר זה יוכל להיסכם רק הערך 1.
להבנתי, אין לכך השלכה על הסיבוכיות, ולכן הצגתי אלגוריתם קצת יותר מפושט – ההרחבה מידית.

שאלה 2

$$P=\begin{array}{c} B\\ K\\ O\\ - \end{array}\begin{pmatrix} 0.1 & 0.325 & 0.25 & 0.325\\ 0.4 & 0 & 0.4 & 0.2\\ 0.2 & 0.2 & 0.2 & 0.4\\ 1 & 0 & 0 & 0 \end{pmatrix}$$
$$\Pr\{l_1=B\}=1$$

a. ההסתברויות נתונות ע"י:

$$\Pr\{Bob\}=Pr\{l_1=B\}\Pr\{l_2=o|l_1=B\}\Pr\{l_3=b|l_2=o\}\Pr\{l_4=-|l_3=b\}$$
$$=1\cdot 0.25\cdot 0.2\cdot 0.325=0.01625$$

$$\Pr\{Koko\}$$
$$=\underbrace{Pr\{l_1=K\}}_{=0}\Pr\{l_2=o|l_1=K\}\Pr\{l_3=k|l_2=o\}\Pr\{l_4=o|l_3=k\}\Pr\{l_5=-|l_4=o\}$$
$$=0\cdot(\cdots)=0$$

$$\Pr\{Bokk\}=Pr\{l_1=B\}\cdot\Pr\{l_2=o|l_1=B\}\cdot\Pr\{l_3=k|l_2=o\}\cdot\underbrace{\Pr\{l_4=k|l_3=k\}}_{=0}$$
$$\cdot\Pr\{l_5=-|l_4=k\}=0\cdot(\cdots)=0$$

$$\Pr\{Boooo\}=Pr\{l_1=B\}\cdot\Pr\{l_2=o|l_1=B\}\cdot\Pr\{l_3=o|l_2=o\}$$
$$\cdot\Pr\{l_4=o|l_3=o\}\cdot\Pr\{l_5=o|l_4=o\}\cdot\Pr\{l_6=-|l_5=o\}$$
$$=1\cdot 0.25\cdot 0.2^3\cdot 0.4=8\cdot 10^{-4}$$

b.

1.

מרחב המצבים: האותיות $S=\{B,K,O,-\}$

מרחב הפעולות: בחירת האות הבאה $A=\{B,K,O,-\}$

במקרה זה יותר נוח להגדיר את ה reward, אותו נרצה למקסם:

$$r(s_k,a_k)=\Pr\{l_{t+1}=a_k|l_t=s_k\}$$
$$R=\prod_{k=1}^{K}r(s_k,a_k)$$

כאשר נדרוש $a_k=-$ לסיום מילה.

2. נניח שישנן N אותיות. עבור כל מיקום במילה k ולכל אות l_k, נחשב מה הצירוף הסביר שנגמר באות זו, ע"פ הקשר

$$R(l_k)=\underset{l_{k-1}}{\operatorname{argmax}}\{R(l_k-1)\Pr\{l_k|l_{k-1}\}\}$$

ונזכור את האות l_{k-1} המביאה למקסימום.

כשהגענו לאות הK, נחשב את האות האחרונה של המילה לפי

$$\hat{l}_K=\underset{l_K}{\operatorname{argmax}}\{R(l_K)\Pr\{l_{K+1}='-'|l_K\}\}$$

בכל שלב עושים N^2 פעולות, וישנם $K+1$ שלבים ולכן סיבוכיות הריצה הינה $O(N^2K)$. לכל מיקום במילה ולכל אות אפשרית, צריך לזכור את האות האופטימלית שהביאה אליו (לצורך בניית המילה הסבירה ביותר), ולכן סיבוכיות הזיכרון הינה $O(NK)$.

3. נגדיר כעת

$$lr(s_k,a_k)=\log\Pr\{l_{t+1}=a_k|l_t=s_k\}$$
$$lR=\log\prod_{k=1}^{K}r(s_k,a_k)=\sum_{k=1}^{K}\log r(s_k,a_k)=\sum_{k=1}^{K}lr(s_k,a_k)$$

וקיבלנו פונקציית מחיר אדיטיבית.

.4
יתרון של פונקציית מחיר כפלית – לא צריך לבצע עיבוד מקדים של ההסתברויות. לא צריך להתעסק עם המקרה המיוחד של הסתברות אפס (הלוגריתם ייתן את הערך $-\infty$, שהמחשב לא 'מכיר').
יתרון של פונקציה אדיטיבית – בפעולת כפל ישנה בעיה של תחום דינאמי. מכיוון שההסתברויות מאוד קטנות, וכופלים הרבה הסתברויות, מהר מאוד מגיעים למספרים מאוד קטנים, שהמחשב כבר לא מסוגל לייצג (גם ב Floating Point!). בחיבור של לוגריתמים, הבעיה הזאת נעשית כמעט זניחה. אם מתכננים מימוש על חומרה ייעודית והמשאבים מוגבלים מאוד – כפל יקר יותר מחיבור (במיוחד אם זו חומרה של fixed point).

.5
הקוד:

```
function [word, reward] = findProbableWord(K)

P = [0.1, 0.325, 0.25, 0.325;...
     0.4, 0,     0.4,  0.2;...
     0.2, 0.2,   0.2,  0.4;...
     1,   0,     0,    0];

dict = {'B','K', 'O'}; % dictionary for P.
numLetters = length(dict);

word_inds = zeros(1,K);

traj = zeros(numLetters,K-1); % remembering the trajectory

% first iteration - last letter was '-'
reward = P(end,1:end-1)';
reward_next = zeros(size(reward));

% Main loop
for i = 1:K-1
    for j = 1:numLetters
        [reward_next(j), traj(j,i)] = max(reward.*P(1:end-1,j));
    end
    reward = reward_next;
end

% last iteration: transition to '_'
[reward, word_inds(K)] = max(reward.*P(1:end-1,end));

% Getting the letters
for i = K-1:-1:1
    word_inds(i) = traj(word_inds(i+1),i);
end
word = [dict{word_inds}];
```

והמילה הכי סבירה:

'BKBKO'
למעשה, הגיוני שזו המילה הכי סבירה – ההסתברויות הכי גבוהות שלא מביאות לסוף מילה הן 0.325 ו-0.4 באותיות B ו-K בהתאמה. נלך הלוך ושוב בין שתי האותיות האלו, ואז מ- K נעבור ל- O בהסתברות 0.4, ונסיים את המילה בהסתברות 0.4 – מעיין בחירה חמדנית בכל רגע. המסקנה: למספר אי זוגי של אותיות, האותיות יהיו B ו-K לסירוגין, כשהאות האחרונה תהיה O (מלבד עבור אורך מילה=1, שאז המילה תהיה B...).

עבור אורך זוגי – קל לראות שבחירה חמדנית שממקסמת את ההסתברות היא זוגות של BK, מלבד הזוג האחרון, שיהיה BO.

שאלה 3

.a

מרחב המצבים : $S = \{(i,j): 1 \le i \le M, 1 \le j \le N\}$, כלומר כל החדרים בהם משה יכול להימצא.

מרחב הפעולות : $A = \{N, E\}$, כאשר N מתאר צעד צפונה, ו-E מתאר צעד מזרחה.
פונקציית המחיר המצטברת : נגיד כי לכל חדר יש 'פרס' של יחידה אם יש בו גבינה, ואחרת אפס :

$$r_k(s_k) = \begin{cases} 1 & s_k\ contains\ cheese \\ 0 & otherwise \end{cases}$$

הפונקציה אותה נרצה למקסם הינה :

$$R = r_0(s_0) + \sum_{k=1}^{K} r_k(s_k)$$

כאשר K הינו מספר הצעדים שמשה עושה ו- $r_0(s_0)$ קובע האם ישנה גבינה בחדר הראשון.

הערה : באופן שקול לחלוטין, ניתן לתאר את הבעיה לפי החדר האחרון וללכת אחורה. אני מעדיף להסתכל על הבעיה באופן סיבתי...

.b

בסופו של דבר, משה ילך $N-1$ צעדים מזרחה ו-$M-1$ צעדים צפונה, ולכן האופק של הבעיה הינו

$$K = (N-1) + (M-1) = N + M - 2$$

.c

מספר המסלולים : מתוך $M + N - 2$ צעדים, צריך לבחור $N-1$ צעדים לכיוון מזרח, ולכן מספר המסלולים יהיה :

$$T(M,N) = \binom{N+M-2}{N-1} = \binom{N+M-2}{M-1}$$

ובפרט עבור $M = 2$:

$$T(2,N) = \binom{N+2-2}{2-1} = \binom{N}{1} = N$$

ההתנהגות פולינומיאלית.

עבור $M = N$:

$$T(N,N) = \binom{N+N-2}{N-1} = \binom{2(N-1)}{N-1}$$

שזה אקספוננציאלי ב-N. (מתנהג בערך כמו 4^N).

.d

1. אם אהרון ומשה יבחרו מסלול אופטימאלי באופן בלתי תלוי, שניהם ילכו באותו מסלול, ואז כל אחד מהם יקבל חצי מהגבינה שהיה מקבל אם הוא היה לבד. אפקטיבית – אחד מהם לוקח את כל הגבינה ומביא חצי לשני, והשני רק הולך בעקבותיו ולא אוסף גבינה – ממש גרוע...

2. מרחב המצבים החדש יכלול מיקום נפרד למשה ולאהרון. כדי לתאר מיקום של אחד מהם, אנחנו זקוקים ל- MN מצבים, ולכן, כדי לתאר את המיקום של שניהם יש צורך ב M^2N^2 מצבים. מרחב הפעולות החדש – כל צירוף של כיוון תנועה של שניהם. כל אחד מהם יכול לנוע בשני כיוונים, ולכן יהיו בסה"כ 4 פעולות : $(N,N), (N,E), (E,N), (E,E)$.

3. באופן דומה, כל מצב יכיל את המיקום של כל אחד מ-K העכברים, ולכן יהיו $N^K M^K$ מצבים. מרחב הפעולות יכיל את כ הצירופים של תזוזות שונות של העכברים, ולכן יהיו 2^K פעולות.

שאלה 4

מעוניינים לפתור:

$$\pi_a^* = argmax_{\pi_a} \min_{\pi_b} R_N(h_N)$$

$$R_N(h_N) = \sum_{k=0}^{N-1} r_k(s_k, a_k, b_k) + R_N(s_N)$$

a. אלגוריתם תכנות דינאמי לבעיה זו:

(1) אתחול:

$$V_N(s) = R_N(s_N)$$

(2) רקורסיה:

$$V_k(s) = \max_{a\in A_k}\left\{\min_{b\in B_k}\left\{r_k(s,a,b) + V_{k+1}\big(f_k(s,a,b)\big)\right\}\right\}$$

(3) מדיניות אופטימאלית:

$$\pi_k(s) \in \operatorname{argmax}_{a\in A_k}\left\{\min_{b\in B_k}\left\{r_k(s,a,b) + V_{k+1}\big(f_k(s,a,b)\big)\right\}\right\}$$

הרעיון הוא אותו רעיון כמו תכנות דינאמי רגיל – בשביל שמסלול יהיה אופטימאלי מזמן k עד זמן N, הוא בהכרח חייב להיות אופטימאלי מזמן $k+1$ ועד זמן N, והבנייה מפה רקורסיבית. במקרה הזה, ה value function מייצגת את הרווח המקסימאלי שניתן להשיג מרגע k ועד הסוף, בהינתן שהתחלנו במצב ה-s והיריב עשה את הפעולות הכי רעות מבחינתנו (והכי טובות מבחינתו).

b.

בכל שלב ברקורסיה, כדי למצוא את המינימום לכל מצב, צריך לעבור על כל הצירופים של a, b, s. מכיוון שיש N שלבים, סיבוכיות הריצה הינה $O(N|S||A||B|)$. סיבוכיות הזיכרון – צריך לזכור את הפעולה המיטבית לכל מצב ולכל דרגה - $O(N|S|)$.

c.

במימוש נאיבי, במשחק איקס-עיגול יש 3^9 מצבים אפשריים (בכל מצב יכול להיות איקס, עיגול או כלום), כל שחקן יכול לעשות 9 פעולות (לבחור אחת מהמשבצות לשים בה איקס או עיגול) ויש לכל היותר 5 תורות עד שהלוח מתמלא. באופן בסיסי, הפרס במשחק – אחד אם ניצחת מינוס אחד אם הפסדת, ואפס אם תיקו (פרס יותר מתוחכם שישאף להאריך את המשחק – פרס 10 על ניצחון, פרס -10 על הפסד, פרס אפס על תיקו, והפרס גדל ב-1 על כל תור שהמשחק נמשך ולא נגמר).

לכן, מספר הפעולות שצריך לעשות הוא מסדר גודל של:

$$3^9 \cdot 9 \cdot 9 \cdot 5 \approx 8 \cdot 10^6$$

שזו כמות פעולות אפשרית לחלוטין.

d.

לא ניתן להשתמש בשיטה זו לשחמט – נתאר את מרחב המצבים באופן מופשט – במשחק שח יש 64 משבצות ו-16 כלים. נניח שמצב מתאפיין רק ע"י בחירת המשבצות שבהן יש כלי (תיאור פשטני שלא אומר אפילו איזה כלי נמצא איפה). עדיין צריך

$$\binom{64}{16} = 500 \cdot 10^{12}$$

מצבים, שזו כמות שאינה ניתנת לעיבוד (וזה בלי להתחשב בכמות הפעולות שניתן לעשות בכל מצב, בעובדה שיכולים להיות פחות כלים על הלוח, בהתחשבות בזהות הכלים וכו').

כלומר – בשח, כמות המצבים והפעולות כל כך גדולה, עד שלא ניתן (מבחינת משאבי עיבוד) לפתור את המשחק בשיטה המוצעת.