Aufbau der Prüfung (in den letzten Jahren)

3h für 80 Punkte

generelle Lernempfehlung: alte Prüfungen, Skript lesen

40 Punkte Moodle Kurzfragen (wie Miniquiz)

einfachster Teil der Prüfung, hier keine Punkte liegen lassen!

Lernempfehlung: Skript lesen & Fragen üben
→ grosses Recap Kahoot
→ Ankis, ich habe meine von letztem Jahr auf meiner Webseite
→ alte Miniquizze

20 Punkte CodeExpert

10 Punkte Flowaufgabe → kennt die verschiedenen Flowtypen / Graphkonstruktionen

10 Punkte Wahrscheinlichkeiten
→ 4-5 Punkte DP → systematisches Lösen (dazu mehr später)

Lernempfehlung: relevante CE-Aufgaben der letzten Wochen nochmal lösen
Polybox mit weiteren alten CE-Aufgaben (s. Webseite)

20 Punkte schriftliche Aufgaben

10 Punkte Wahrscheinlichkeiten → sehr häufig Schranken

10 Punkte Beweise

Lernempfehlung: Beweise im Skript lesen und verstehen
falls ihr viel Zeit habt: Polybox mit alten Übungsaufgaben (auch auf meiner Webseite)

1.1 Wahrscheinlichkeitsaufgabe

Sei $n \geq 2$ ganzzahlig und eine gerade Zahl. Betrachten Sie ein Gitter der Grösse n x n - d.h. ein Gitter auf n x n Knoten - und nehmen Sie weiter an, dass jede seiner Kanten unabhängig mit Wahrscheinlichkeit $\frac{1}{2}$ blau - und ansonsten rot gefärbt wird. Wir nennen eine Zelle einfarbig, falls alle vier sie umschliessenden Kanten dieselbe Farben haben, d.h. alle sind entweder blau oder alle sind rot.

1. Sei X die Gesamtzahl einfarbiger Zellen. Zeigen Sie, dass $E[X] = \frac{(n-1)^2}{8}$.

   Im verbleibenden Teil wollen wir zeigen, dass mit Wahrscheinlichkeit, die gegen 1 strebt, wenn n gegen unendlich geht, die Anzahl einfarbiger Zellen mindestens $\frac{(n-1)^2}{12}$ beträgt.

2. Um dies zu beweisen schlägt Ihre Freundin Alice folgendes Argument vor:

   "X ist eine Summe von $(n-1)^2$ Indikatorvariablen, wir können also die Ungleichung von Chernoff verwenden und daraus ableiten, dass $Pr[X \leq \frac{(n-1)^2}{12}] \leq e^{-\frac{1}{2}\cdot\frac{1}{3}^2\cdot\frac{(n-1)^2}{3}} \to 0$." Erklären Sie, weshalb das Argument von Alice fehlerhaft ist.

3. Reparieren Sie das Argument von Alice und zeigen Sie, dass die Schlussfolgerung $Pr[X \leq \frac{(n-1)^2}{12}] \to 0$ dennoch zutrifft.

1) $X_{i,j} = \begin{cases} 1 & \text{falls Zelle } i,j \text{ einfarbig} \\ 0 & \text{sonst} \end{cases}$ $\quad X := \#\text{einfarbige Zellen}$

$Pr[|X - E[X]| \geq t] \leq \frac{Var[X]}{t^2}$

$E[\sum_{i,j \leq (n-1)} X_{i,j}] \overset{\text{Linearität von E}}{=} \sum_{i,j \leq n-1} E[X_{i,j}] = \frac{(n-1)^2}{8}$

2) $X_{i,j}$ nicht unabhängig

3) $Pr[X \leq \frac{(n-1)^2}{12}] = Pr[X - E[X] \leq -\frac{(n-1)^2}{24}] \leq Pr[|X - E[X]| \geq \frac{(n-1)^2}{24}] \leq \frac{7(n-1)^2}{2^6} = \frac{7(n-1)^2 \cdot 24^2}{2^6 \cdot (n-1)^4 2} \xrightarrow{n\to\infty} 0$

$Var[X] = E[X^2] - E[X]^2$

$E[X^2] = E[(\sum_{i,j \leq (n-1)} X_{i,j})^2] = E[\sum_{\substack{i,j,k,\ell \\ \leq (n-1)}} X_{i,j} X_{k,\ell}] \quad i=k,\ j=\ell$

① teilen alle Kanten tritt $(n-1)^2$-mal auf $p = \frac{1}{8}$

② teilen (k) keine Kante tritt $(n-1)^4 - (n-1)^2$-mal auf $p = \frac{1}{2^6} = \frac{1}{64}$ $\quad \frac{1}{2^6}$

$Var[X] = E[X^2] - E[X]^2 = \frac{(n-1)^2}{8} + \frac{(n-1)^4 - (n-1)^2}{2^6} - \frac{(n-1)^4}{2^6}$

$= \frac{7(n-1)^2}{2^6}$

1.2 Beweisaufgaben

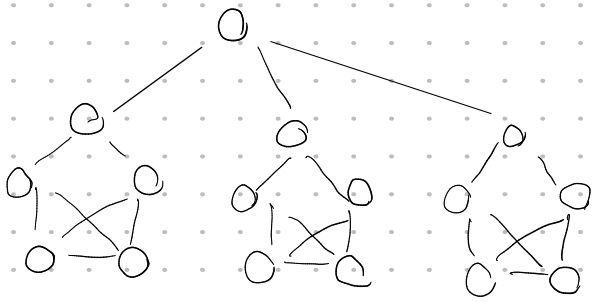
Finden Sie für jede Aussage entweder einen Beweis oder formulieren Sie ein Gegensbeispiel.

1. Der Satz von Dirac ist optimal. Für jedes $n \geq 3$ gibt es einen Graphen G mit Minimalgrad $\lceil \frac{n}{2} \rceil - 1$, der keinen Hamiltonkreis enthält. (FS21 E2)
2. Sei G ein 3-regulärer Graph, dann enthält G ein perfektes Matching.
3. Wenn es eine Mange $K \subseteq V$ gibt, sodass für alle $v \in K$ gilt, dass $deg(v) \leq k$ und für alle anderen Knoten $w \in V$ gilt $deg(w, V \setminus K) \leq k$, wobei $deg(w, V \setminus K)$ die Anzahl der Kanten vom Knoten w in die Menge $V \setminus K$ ist, dann kann man G mit $k+1$ Farben färben. (FS21 E4)

1) $G = (A \uplus B, E)$ $\quad |A| = \lceil \frac{n}{2} \rceil - 1$ $\quad |B| = n - |A| = \lfloor \frac{n}{2} \rfloor + 1$

$E = \{ \{u, v\} \mid u \in A, v \in B \}$ $\quad |A| \neq |B| \longrightarrow$ kein Hamiltonkreis

2)



2 Wahrscheinlichkeitsaufgabe CodeExpert: Video Game

Du spielst ein Videospiel mit $n+1$ Levels, nummeriert von 0 bis n. Du beginnst bei Level 0, und das Spiel verläuft in Runden.
In jeder Runde spielst du das Level, auf dem du dich befindest, und je nachdem, wie gut du abschneidest, kannst du in der nächsten Runde auf ein höheres Level aufsteigen oder in ein niedrigeres absteigen.
Wenn du ein Level i *ausgezeichnet* meisterst, steigst du um x Level auf, also direkt zu Level $i+x$.
Wenn du ein Level i *knapp bestehst*, steigst du um y Level auf, also zu Level $i+y$, wobei $y < x$ ist.
Wenn du ein Level i *nicht bestehst*, fällst du um ein Level zurück, also zu Level $i-1$.

Für jedes Level $i \in \{0, \ldots, n\}$ gilt: Du bestehst es entweder mit *ausgezeichnet* mit Wahrscheinlichkeit e_i, *knapp* mit Wahrscheinlichkeit b_i, oder du *fällst durch* mit Wahrscheinlichkeit f_i.
Level 0 ist sehr einfach, daher kann man es nicht nicht bestehen. Wenn du insgesamt z-mal durchfällst, ist das Spiel vorbei. Wenn du dich auf einem Level $i > n-x$ befindest und es ausgezeichnet meisterst oder auf einem Level $i > n-y$ und es knapp bestehst (oder ausgezeichnet meisterst), ist das Spiel ebenfalls vorbei.

1. Wie hoch ist die Wahrscheinlichkeit, dass du nach zwei Runden auf Level $x+y$ landest? (3 Punkte)
2. Unter der Bedingung, dass du in der zweiten Runde durchgefallen bist, wie groß ist die Wahrscheinlichkeit, dass du das Level in der ersten Runde *ausgezeichnet* gemeistert hast? (3 Punkte)
3. Wie viele Runden wirst du im Mittel spielen, bevor das Spiel vorbei ist? (4 Punkte)

Input:

Die erste Zeile der Eingabedatei enthält eine Zahl $t \leq 30$, die die Anzahl der Testfälle angibt. Jeder der t Testfälle ist wie folgt beschrieben:

- Er beginnt mit einer Zeile, die zwei ganze Zahlen n und q enthält, getrennt durch ein Leerzeichen. Dabei gibt n den Index des letzten Levels im Spiel an ($2 \leq n \leq 4000$) und q die Art der Frage, die beantwortet werden soll ($1 \leq q \leq 3$).
- Die nächste Zeile enthält drei ganze Zahlen x, y und z, getrennt durch Leerzeichen. Diese geben an: die Anzahl an Levels, die man voranschreitet, wenn man hervorragend ist ($2 \leq x \leq 10$), die Anzahl an Levels, die man voranschreitet, wenn man knapp besteht ($1 \leq y < x$), und die Anzahl an Fehlversuchen, die zum Spielende führen ($1 \leq z \leq 800$).
- Die i-te der folgenden $n+1$ Zeilen ($0 \leq i \leq n$) enthält drei reelle Zahlen e_i, b_i, f_i, getrennt durch Leerzeichen. Es ist garantiert, dass $f_0 = 0$ and $e_i + b_i + f_i = 1$.

Bei Wahrscheinlichkeits-DP wollen wir meist eine der folgenden Dinge für die Rekursion nutzen:

a) $Pr[X] = \sum_{i=1}^{n} Pr[X|A_i] \cdot Pr[A_i]$

b) $E[X] = \sum_{i=1}^{n} E[X|A_i] \cdot Pr[A_i]$

$\bigcup_{i=1}^{n} A_i = \Omega$
$A_i \cap A_j = \emptyset$ für $i \neq j$

```java
class Main {
    public static void main(String[] args) {
        // Uncomment this line if you want to read from a file
        In.open("public/custom.in");
        Out.compareTo("public/custom.out");

        int t = In.readInt();
        for (int i = 0; i < t; i++) {
            testCase();
        }

        // Uncomment this line if you want to read from a file
        In.close();
    }
    static double[][] dp;
    static double[] e;
    static double[] f;
    static double[] b;
    static int n;
    static int x;
    static int y;
    static int z;

    public static void testCase() {
        // Input using In.java class
        n = In.readInt();
        int q = In.readInt();
        x = In.readInt();
        y = In.readInt();
        z = In.readInt();
        e = new double[n+1];
        b = new double[n+1];
        f = new double[n+1];
        for(int i=0; i<=n; i++) {
          e[i] = In.readDouble();
          b[i] = In.readDouble();
          f[i] = In.readDouble();
        }
        if(q==1) Out.println(task1());
        else if(q==2) Out.println(task2());
        else Out.println(task3());
    }

    public static double task1(){
      return e[0] * b[x] + b[0] * e[y];
    }

    public static double task2(){
      double temp = e[0] * f[x];
      return temp/(temp + b[0] * f[y]);
    }

    public static double task3(){
      //dp[i][j] = erwartete Anzahl an Runden bis wir das Spiel beenden, wenn wir bei Level i sind und j-mal gefailed haben
      dp = new double[n+1][z+1];
      for(int i=0; i<=n; i++){
        for(int j=0; j<=z; j++) dp[i][j] = -1;
      }
      return solve(0,0);
    }

    public static double solve(int i, int j){
      if(i > n || i < 0 || j >= z) return 0;
      if(dp[i][j] != -1) return dp[i][j];
      dp[i][j] = e[i] * (1 + solve(i+x, j)) + b[i] * (1 + solve(i+y, j)) + f[i] * (1 + solve(i-1, j+1));
      return dp[i][j];
    }
}
```