ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing Group

HS 2009

Prof. Dr. R. Wattenhofer
Raphael Eidenbenz
Jasmin Smula

Discrete Event Systems
Solution to Exercise 14

1 Kompetitive Weihnachten

a) Falls unser Algorithmus eine Tanne der Grösse c kauft, kommt danach im schlechtesten Fall eine Tanne der Grösse M, was eine kompetitive Ratio von $\rho_A = \frac{M}{c}$ ergibt. Um ρ_A möglichst klein zu halten, sollte Roger nur dann eine Tanne kaufen, wenn dessen Grösse c möglichst gross ist. Verschmäht nun aber Roger eine Tanne der Grösse $c - \epsilon$ in der Hoffnung noch eine grössere Tanne zu finden, kommen im schlimmsten Fall bloss noch Tannen der Grösse m, was zu einer kompetitiven Ratio von $\rho_B = \frac{c-\epsilon}{m}$ führt. In diesen beiden Fällen soll ρ möglichst klein sein, so dass Roger eine möglichst grosse Tanne kauft. D.h. wir müssen die Schranke c, oberhalb welcher Roger eine Tanne kauft, so wählen, dass ρ_A und ρ_B minimiert sind. Dies ist der Fall für $\rho_A = \rho_B$. Wir erhalten als Kaufschranke $c = \sqrt{mM}$ und die zugehörige kompetitive Ratio ist $\sqrt{\frac{M}{m}}$.

b) Dies wurde bereits unter a) besprochen.

c) Wir nehmen an, der Verkäufer könne immer noch wählen, welche Tanne er Roger an der i-ten Stelle präsentiert, wenn er diesen auf sich zu kommen sieht. Konkret nehmen wir an, er könne die Grösse der i-ten Tanne wählen in Abhängigkeit davon, ob Roger bis dahin schon eine gekauft hat oder nicht. Unter dieser Annahme hilft Randomisierung bei diesem Beispiel nichts. Der Adversary kann mit folgender Input Sequenz die obige kompetitive Ratio erzwingen: $c\ c\ c\ c\ c\ \ldots\ c\ c\ c\ \{M|m\}$. Falls der Algorithmus einen Baum der Grösse $c = \sqrt{mM}$ kauft, hat die letzte Tanne die Höhe M, ansonsten m.

2 Die beste Sekretärin

a) Um eine ungünstigste Reihenfolge der Sekretärinnen zu vermeiden, bestellen wir sie in einer zufälligen Reihenfolge herein. Bei der ersten Hälfte ($\frac{n}{2}$ Leute) schauen wir bloss wie gut die Bewerber sind und merken uns die beste Güte G. In der zweiten Hälfte nehmen wir eine Sekretärin falls ihre Güte G übersteigt.

Die Wahrscheinlichkeit, dass die zweitbeste Sekretärin in der ersten Hälfte ist, ist 0.5. Die Wahrscheinlichkeit, dass sich die beste Sekretärin in der zweiten Hälfte befindet ist ebenfalls 0.5. Falls die Zweitbeste sich in der ersten Hälfte und die Beste sich in der zweiten Hälfte befindet, wählen wir tatsächlich die Beste. Dieser Fall tritt mit einer Wahrscheinlichkeit von 0.25 ein.

b) Die Analyse für den obigen Algorithmus ist natürlich eine sehr grobe Abschätzung. Zudem haben wir ohne zu Begründen angenommen, dass der Algorithmus zuerst *die Hälfte* der Leute anschaut, und erst dann auswählt. Zudem haben wir bloss den Spezialfall angeschaut, dass die zweitbeste Sekretärin im ersten Teil war und die Beste im zweiten Teil. Wir hätten ebenfalls Erfolg wenn die drittbeste im ersten Teil ist, und die Beste und die Zweitbeste im zweiten Teil, und die Beste vor der Zweitbesten kommt.

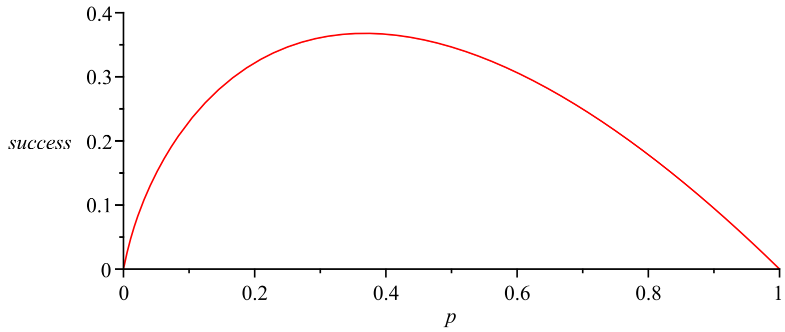
Für die genaue Analye nehmen wir nun an, dass der Algorithmus zuerst $p \cdot n$ Leute anschaut (mit $p \in [0..1]$), und dann jene Sekretärin wählt, welche besser als alle zuvor ist.

Der erste Fall, dass die Zweitbeste im ersten Teil kommt und die Beste im zweiten Teil hat eine Wahrscheinlichkeit von $p_2 = p \cdot (1-p)$.

p_3 ist die Wahrscheinlichkeit dass die Drittbeste im ersten Teil kommt, und die Beste und Zweitbeste im zweiten Teil, in dieser Reihenfolge. Wir haben dass $p_3 = p \cdot (1-p)^2 \cdot \frac{1}{2}$. Der letzte Term ist die Wahrscheinlichkeit, dass die Beste und die Zweitbeste auch wirklich in dieser Reihenfolge eintreffen. Generell suchen wir p_k, die Wahrscheinlichkeit, dass die k-te Beste im ersten Teil kommt, und die $1 \dots (k-1)$ Besten im zweiten Teil, wobei die Beste zuerst kommen muss. Ähnlich wie zuvor[1] haben wir dass $p_k = p \cdot (1-p)^{k-1} \cdot \frac{1}{k-1}$.

Somit können wir die Wahrscheinlichkeit für eine erfolgreiche Wahl berechnen mit

$$\begin{aligned} \mathbb{P}_{success} &= \sum_{i=2}^{\infty} p_i & (1) \\ &= \sum_{i=1}^{\infty} \frac{p}{i} \cdot (1-p)^i & (2) \\ &= p \cdot \sum_{i=1}^{\infty} \frac{(1-p)^i}{i} & (3) \end{aligned}$$



Diese Funktion erreicht ein Maximum für $p = \frac{1}{e}$. Unser Algorithmus sollte also bloss $\frac{1}{e} \approx 36.8\%$ der Leute anschauen, und dann bereits mit der Wahl beginnen. Die so erreichte Erfolgswahrscheinlichkeit ist (ebenfalls!) $\frac{1}{e}$.

[1] Die Wahrscheinlichkeiten sind eigentlich nicht unabhängig voneinander, aber für grosse n lässt sich dies vernachlässigen.