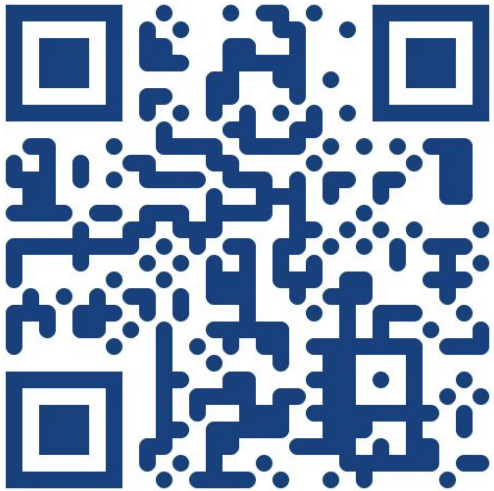
Algorithms and Probability

Exercise Session 9



https://n.ethz.ch/~ahmala/anw

Algorithmen und Wahrscheinlichkeit
Theorie-Aufgaben 7

Abgabe in Moodle () bis zum 25.04.2024 um 10:00 Uhr.

Aufgabe 1 – *Couch to* k *k*

Endlich ist der Frühling da in deiner Heimatstadt. Die Sonne scheint - und Blumen spriessen zufällig verteilt am Strassenrand. Bei diesem schönen Wetter entscheidest du dich, zusammen mit deiner Französischen Bulldogge Doug die diesjährige Spaziersaison einzuläuten. Es gibt nur ein Problem - die Blumen sind zwar schön, zu viele auf einmal führen jedoch dazu, dass Dougs Heuschnupfen ausbricht, so dass eure Spazier-Route deutlich weniger angenehm ist. Ein paar wenige Blumen sind für Doug gerade noch auszuhalten, wenn allerdings mindestens $\frac{3}{4}$ aller Strassen auf eurer Route von Blumen gesäumt sind, ist der Spaziergang *kein Vergnügen* mehr. Wenn an allen Strassen auf eurer Route Blumen wachsen, so *schnieft* Doug für den Rest des Tages.

Wir nehmen an, dass deine Heimatstadt - wie so viele Heimatstädte - ein Graph $G = (V, E)$ ist. Die Knoten sind dabei Kreuzungen und die Kanten sind Strassen, welche zwei Kreuzungen verbinden. Die Anfangsposition eurer Spazier-Route ist irgend eine Kreuzung $v \in V$ und die Route ist ein gegebener Kreis $C = (v_0 = v, v_1, v_2, \ldots, v_{k-1}, v_k = v)$ der Länge k.

Bei jeder Strasse $e \in E$ gibt es Blumen am Strassenrand mit Wahrscheinlichkeit $\frac{1}{2}$. Dies geschieht unabhängig für jede Strasse.

(a) Berechne, unter Zuhilfenahme der Ungleichung von Markov, eine obere Schranke für die Wahrscheinlichkeit, dass der Spaziergang kein Vergnügen ist.

(b) Berechne, unter Zuhilfenahme der Ungleichung von Chebyshev, eine obere Schranke für die Wahrscheinlichkeit, dass der Spaziergang kein Vergnügen ist.

(c) Nimm an, dass du $n-1$ Freunde mit je einem vom Heuschnupfen geplagten Hund hast, welche am selben Tag wie du spazieren gehen wollen. Ihr alle habt eure individuellen Spazier-Routen, welche aber nicht disjunkt sein müssen. Berechne die erwartete Anzahl *schniefender* Hunde, nachdem ihr alle eure Spaziergänge beendet habt. Zeige, falls gilt $k \geq \log_2(n) + 1$, dann gibt es mit Wahrscheinlichkeit mindestens $\frac{1}{2}$ keinen einzigen schniefenden Hund.

(d) Nimm an, dass $k = 1000 \log_2 n$ und $n \geq 2$. Zeige, dass mit Wahrscheinlichkeit mindesten 0.99 alle Spaziergänge ein Vergnügen sein werden.

Exam Problem from FS 2020

1. Wir werfen eine faire Münze $n \geq 1$ mal. Sei X die Anzahl der Würfe bei denen die Münze 'Kopf' zeigt. Geben Sie möglichst gute obere Schranken für $\Pr[X \geq 0.75n]$ an.

i) Mithilfe der Ungleichung von Markov. (1 Punkt)

ii) Mithilfe der Ungleichung von Chebychev. (2 Punkte)

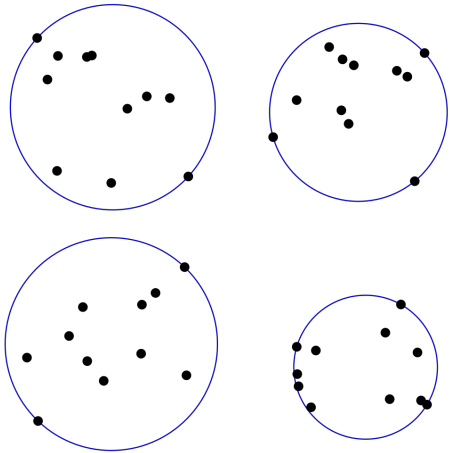
iii) Mithilfe der Chernoff Schranken. (2 Punkte)

Altogether: 5/44 Punkten $\approx$ 11%

Minimum(Smallest) Enclosing Circle

the smallest enclosing circle is unique

for any set P with $|P| \geq 3$, there exists a subset $Q \subseteq P$ with $|Q| = 3$ such that $C(P) = C(Q)$. The points in Q determine C uniquely.



Naive Algorithm

```
1  for all Q ∈ (P choose 3) do
2      compute C(Q)
3      if P ⊆ C(Q)
4          return C(Q)
```

Las Vegas Algorithm

```
1  while true
2      pick Q ∈ (P 11) uniformly at random
3      compute C(Q)
4      if P ⊆ C(Q)
5          return C(Q)
6      double all points outside C(Q)
```

https://en.wikipedia.org/wiki/Smallest-circle_problem#Welzl's_algorithm

Chance and (lack of) plans

Since 1996, Emo Welzl has been a professor at the Institute of Theoretical Computer Science. He has dedicated his career to developing and analysing combinatorial algorithms and discrete mathematics. Among other things, he has worked on algorithmic geometry and the analysis of geometric structures. As he will be retiring at the end of January, we look back on his career in this interview and learn how he found his way into theoretical computer science and what fascinates him about it.

30.01.2024 by Anna Janka

en.wikipedia.org/wiki/Smallest-circle_problem#Welzl's_algorithm

Contents hide

their convex hull, it is unconstrained solution, otherwise the direction to the nearest edge determines the half-plane of the unconstrained solution.)

Welzl's algorithm [edit]

The algorithm is recursive.

The initial input is a set P of points. The algorithm selects one point p randomly and uniformly from P, and recursively finds the minimal circle containing $P – \{p\}$, i.e. all of the other points in P except p. If the returned circle also encloses p, it is the minimal circle for the whole of P and is returned.

Otherwise, point p must lie on the boundary of the result circle. It recurses, but with the set R of points known to be on the boundary as an additional parameter.

The recursion terminates when P is empty, and a solution can be found from the points in R: for 0 or 1 points the solution is trivial, for 2 points the minimal circle has its center at the midpoint between the two points, and for 3 points the circle is the circumcircle of the triangle described by the points. (In three dimensions, 4 points require the calculation of the circumsphere of a tetrahedron.)

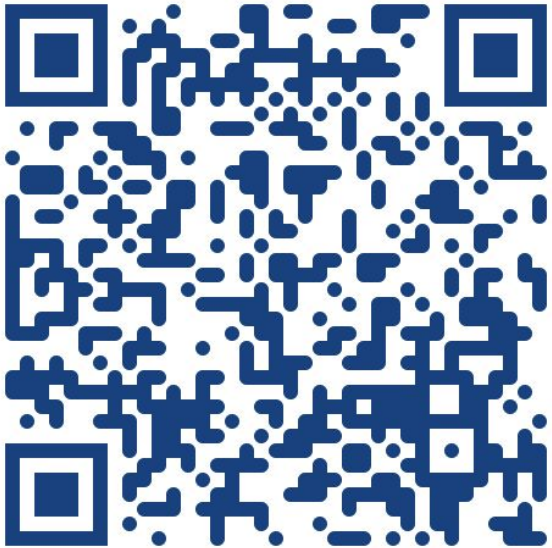
Recursion can also terminate when R has size 3 (in 2D, or 4 in 3D) because the remaining points in P must lie within the circle described by R.

```
algorithm welzl is[8]
    input: Finite sets P and R of points in the plane |R| ≤ 3.
    output: Minimal disk enclosing P with R on the boundary.

    if P is empty or |R| = 3 then
        return trivial(R)
    choose p in P (randomly and uniformly)
    D := welzl(P – {p}, R)
    if p is in D then
        return D

    return welzl(P – {p}, R ∪ {p})
```

In Class Exercise



https://n.ethz.ch/~ahmala/anw/material/smallest_circle.pdf

Finding Duplicates

Given an array of integers.

How to find the duplicates?

Finding Duplicates II

Given a list of strings.

How to find the duplicates?

Sorting sometimes not possible

List Elements are large

=> Comparison is expensive

=> Memory Access is expensive

Hashing

Hash function h(k) : key -> index

h(string): string -> integer

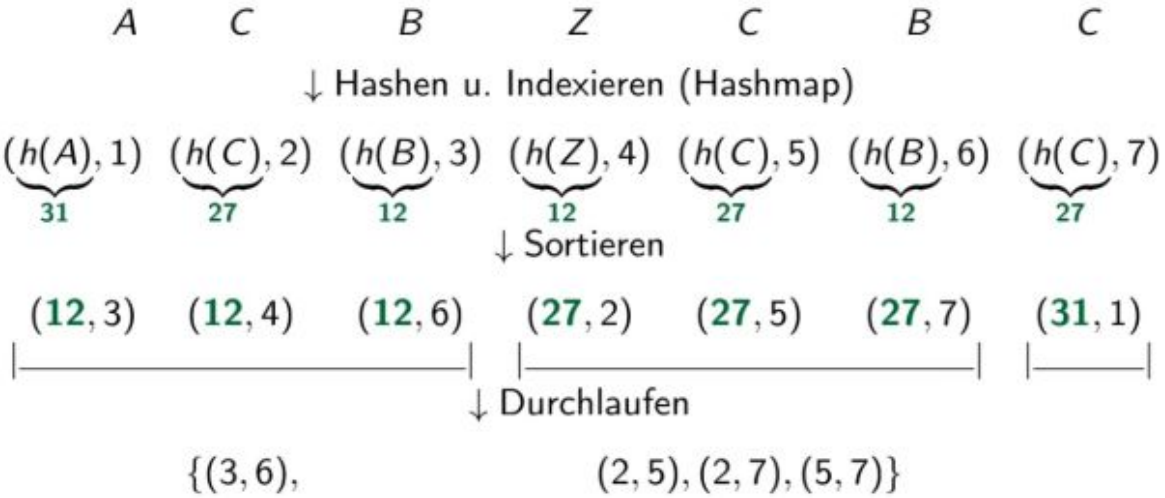
Hashing

h is efficiently computable

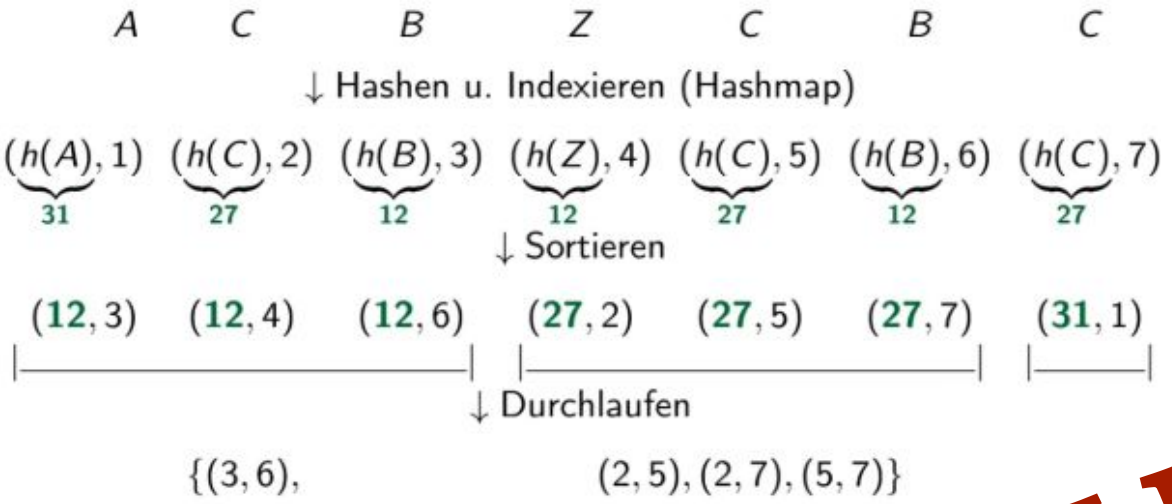
h acts like a random variable

$$\forall u \in U \ \forall i \in [m] : \Pr[h(u) = i] = \frac{1}{m}$$

Hashing


A C B Z C B C
↓ Hashen u. Indexieren (Hashmap)
(h(A), 1) (h(C), 2) (h(B), 3) (h(Z), 4) (h(C), 5) (h(B), 6) (h(C), 7)
31 27 12 12 27 12 27
↓ Sortieren
(12, 3) (12, 4) (12, 6) (27, 2) (27, 5) (27, 7) (31, 1)
↓ Durchlaufen
{(3, 6), (2, 5), (2, 7), (5, 7)}

Hashing



COLLISIONS

Let's find the probability of a collision😊

Given n strings s_1, s_2, ..., s_n

$$\forall u \in U \ \forall i \in [m] : \Pr[h(u) = i] = \frac{1}{m}$$

(i, j), 1 <= i, j <=n,

If s_i and s_j are not the same but h(s_i) = h(s_j) holds then we have a collision

Find an upper bound for the expected number of collisions(assume all strings are different for your convenience)

Let's find the probability of a collision😊

Given n strings s_1, s_2, ..., s_n

$$\forall u \in U \ \forall i \in [m] : \Pr[h(u) = i] = \frac{1}{m}$$

(i, j), 1 <= i, j <=n,

If s_i and s_j are not the same but h(s_i) = h(s_j) holds then we have a collision

Find an upper bound for the expected number of collisions(assume all strings are different for your convenience)

Bloom Filters

k hash functions

boolean array M[1,...,m] with size m

- Hash the element with k hash functions
- Check if all corresponding bits in M are 1.
  - If any are 0, the element is definitely not in the set.
  - If all are 1, the element is probably in the set (but there could be false positives).
- Set the bits at the resulting indices in M to 1.

Bloom Filters - Example

$$\mathcal{S} = (\underset{1}{A}\ \underset{2}{C}\ \underset{3}{B}\ \underset{4}{Z}\ \underset{5}{F}\ \underset{6}{H}\ \underline{\underset{7}{C}}\ )$$

$m = 9,\ k = 3.$

$$A \mapsto (5,4,1), B \mapsto (4,6,1), C \mapsto (5,1,2),$$
$$F \mapsto (1,9,4), H \mapsto (4,7,5), Z \mapsto (4,2,5)$$

$$M = (\underset{1}{1}\ \underset{2}{1}\ \underset{3}{0}\ \underset{4}{1}\ \underset{5}{1}\ \underset{6}{1}\ \underset{7}{1}\ \underset{8}{0}\ \underset{9}{1}\ )$$

$$\mathcal{L} = \{4,7\}$$

Bloom Filters - Example

$$\mathcal{S} = (\underset{1}{A}\ \underset{2}{C}\ \underset{3}{B}\ \underset{4}{Z}\ \underset{5}{F}\ \underset{6}{H}\ \underline{\underset{7}{C}})$$

$m = 9,\ k = 3.$

$$A \mapsto (5,4,1), B \mapsto (4,6,1), C \mapsto (5,1,2),$$
$$F \mapsto (1,9,4), H \mapsto (4,7,5), Z \mapsto (4,2,5)$$

$$M = (\underset{1}{1}\ \underset{2}{1}\ \underset{3}{0}\ \underset{4}{1}\ \underset{5}{1}\ \underset{6}{1}\ \underset{7}{1}\ \underset{8}{0}\ \underset{9}{1})$$

$$\mathcal{L} = \{4,7\}$$

COLLISIONS 😭

Let's find the probability of a collision😊

$$\forall u \in U \ \forall i \in [m] : \Pr[h(u) = i] = \frac{1}{m}$$

Given k hash functions and a boolean array M[1,2,...,m]

Find an upper bound for the expected number of collisions(assume all strings are different for your convenience)

During the insertion of the i-th element you get a collision iff M[h_j(s_i)] = 1 for all 1<=j<=k

First, find an upper bound for the probability of getting a collision during the process of s_i.

What happens if we increase the values of k and m?