Automatically generated, personalized Exams (and their Solutions)

Prof. Dr. Jens Dittrich
Big Data Analytics Group
FR Informatik
Saarland Informatics Campus

Motivation

C.

How to inhibit Plagiarism?

Personal exams!

How we did it:

Compilation command: **SEED := f(MATRICULATIONID)**

```
final-exam — -zsh — 124×47
jedi@mhdhcp08 final-exam % MATRICULATIONID=1234 CWD="." SEED=42 pdflatex --enable-pipes --shell-escape final-exam.tex
```

LaTeX Source:

```
\begin{small}
    \verbatiminput{|"python3 ${CWD}/python/dynprog.py ${SEED} True False"}
\end{small}
```

python:

```
if __name__ == "__main__":
    if len(sys.argv) < 2:
        raise ValueError("specify exactly one parameter with the seed")

    seed = int(sys.argv[1])
    dynProg(seed, dist.strtobool(sys.argv[2]), dist.strtobool(sys.argv[3]))
```

```
def dynProg(seed, showProblem=False, showSolution=False, outdir=''):
    rand.seed(seed)

    from collections import namedtuple
    relations = {'A', 'B', 'C'}
    numberOfRelations = len(relations)

    cardinalities = {'A': rand.randint(
        45, 60)*10, 'B': rand.randint(25, 40)*10, 'C': rand.randint(40, 60)*10}

    def printCardinalities():
        for idx, key in enumerate(cardinalities):
            if idx > 0:
                print(', ', sep='', end='')
            print('|', key, '| = ', cardinalities[key], sep='', end='')
    joinSelectivities = {}

    class JoinPredicate:
        def __init__(self, left, right, joinSelectivity):
            self.left = left
            self.right = right
            self.joinSelectivity = joinSelectivity

        def __repr__(self):
            return "Join("+self.left+", "+self.right+"): sel="+str(self.joinSelect

    joinPredicates = [
        JoinPredicate('A', 'B', round(rand.uniform(0.01, 0.07), 2)),
        JoinPredicate('A', 'C', round(rand.uniform(0.02, 0.07), 2)),
        JoinPredicate('B', 'C', round(rand.uniform(0.03, 0.09), 2))
    ]
```

Compilation command: **SEED** := f(MATRICULATIONID)

```
final-exam — -zsh — 124×47
jedi@mhdhcp08 final-exam % MATRICULATIONID=1234 CWD="." SEED=42 pdflatex --enable-pipes --shell-escape final-exam.tex
```

LaTeX Source for the problem section:

```
\begin{small}
    \verbatiminput{|"python3 ${CWD}/python/dynprog.py ${SEED} True False"}
\end{small}
```

pdf:

Database Systems WiSe 2020/2021
Big Data Analytics Group
Matriculation number: 1234

Prof. Dr. Jens Dittrich
Final Exam
February 24th 2021

SAARLAND UNIVERSITY
COMPUTER SCIENCE

2 Dynamic Programming (29 Points)

Assume you want to compute the following query:

```
SELECT A.a, B.b, C.c
FROM   A, B, C
WHERE  A.k=B.z AND A.s=C.m AND B.d=C.e
```

You have no indexes available and you are given the following relations with their cardinalities as well as the following join predicates and their join selectivities:

For simplicity you can assume that the selectivities are independent and need to be multiplied if sub-problems can be joined by multiple join predicates.

```
cardinalities:
|A| = 480, |B| = 250, |C| = 480

join predicates & selectivities:
Join(A, B): sel=0.02
Join(A, C): sel=0.03
Join(B, C): sel=0.04
```

LaTeX Source for the solution section:

```
\begin{small}
\input{|"echo 'matriculation:' $MATRICULATIONID"}
\verbatiminput{|"python3 ${CWD}/python/dynprog.py ${SEED} False True"}
\end{small}
```

pdf:

Solution:

For each subproblem of size 2: 2 points for optimal costs, 2 points for optimal plan, 1 point for correct output size
For the final subproblem of size 3: 3.5 points for optimal costs, 3.5 points for optimal plan, 3 points for correct output size
matriculation: 1234

```
% seed: 42

problem size to solve: 2
AB
 c=830 pl=SHJ(SCAN(B),SCAN(A)) crd=2400
 c=803 pl=PJ(SCAN(B),SCAN(A)) crd=2400
 c=922 pl=SHJ(SCAN(A),SCAN(B)) crd=2400
 c=803 pl=PJ(SCAN(A),SCAN(B)) crd=2400
BC
 c=830 pl=SHJ(SCAN(B),SCAN(C)) crd=4800
 c=803 pl=PJ(SCAN(B),SCAN(C)) crd=4800
 c=922 pl=SHJ(SCAN(C),SCAN(B)) crd=4800
 c=803 pl=PJ(SCAN(C),SCAN(B)) crd=4800
AC
 c=1152 pl=SHJ(SCAN(A),SCAN(C)) crd=6912
 c=1056 pl=PJ(SCAN(A),SCAN(C)) crd=6912
 c=1152 pl=SHJ(SCAN(C),SCAN(A)) crd=6912
 c=1056 pl=PJ(SCAN(C),SCAN(A)) crd=6912

problem size to solve: 3
ABC
 c=8318 pl=SHJ(SCAN(B),PJ(SCAN(A),SCAN(C))) crd=1382
 c=8934 pl=PJ(SCAN(B),PJ(SCAN(A),SCAN(C))) crd=1382
 c=6275 pl=SHJ(SCAN(A),PJ(SCAN(B),SCAN(C))) crd=1382
 c=6611 pl=PJ(SCAN(A),PJ(SCAN(B),SCAN(C))) crd=1382
 c=3875 pl=SHJ(SCAN(C),PJ(SCAN(B),SCAN(A))) crd=1382
```

one personal exam for each student

identical text

variation

automatically generated solution (hidden)

Variant 1

Matriculation number: 1234

2 Dynamic Programmi

Assume you want to compute the follov

```
SELECT A.a, B.b, C.c
FROM   A, B, C
WHERE  A.k=B.z AND A.s
```

You have no indexes available and you a
the following join predicates and their j

For simplicity you can assume that the
problems can be joined by multiple join

```
cardinalities:
|A| = 520, |B| = 390, |C| = 480

join predicates & selectivities:
Join(A, B): sel=0.01
Join(A, C): sel=0.06
Join(B, C): sel=0.08
```

```
final best plan:
c=3701 pl=SHJ(SCAN(C),PJ(SCAN(A),SCAN(B))) crd=4672
```

Variant 2

Matriculation number: 2345

2 Dynamic Programmi

Assume you want to compute the follc

```
SELECT A.a, B.b, C.c
FROM   A, B, C
WHERE  A.k=B.z AND A.
```

You have no indexes available and you
the following join predicates and their

For simplicity you can assume that tl
problems can be joined by multiple joi

```
cardinalities:
|A| = 590, |B| = 270, |C| = 560

join predicates & selectivities:
Join(A, B): sel=0.02
Join(A, C): sel=0.05
Join(B, C): sel=0.03
```

```
final best plan:
c=4916 pl=SHJ(SCAN(C),PJ(SCAN(B),SCAN(A))) crd=2676
```

Variant 3

Matriculation number: 3456

2 Dynamic Programmi

Assume you want to compute the follc

```
SELECT A.a, B.b, C.c
FROM   A, B, C
WHERE  A.k=B.z AND A.
```

You have no indexes available and you
the following join predicates and their

For simplicity you can assume that tl
problems can be joined by multiple joi

```
cardinalities:
|A| = 460, |B| = 330, |C| = 460

join predicates & selectivities:
Join(A, B): sel=0.06
Join(A, C): sel=0.04
Join(B, C): sel=0.04
```

```
final best plan:
c=7585 pl=SHJ(SCAN(A),PJ(SCAN(B),SCAN(C))) crd=6703
```

Serving & Scaling

Basically two solutions:

1.) on demand creation (good enough for small exams)

2.) precompute (for better scalability)

Plus (if required):

- horizontal partitioning

- timeshifting, e.g. some students start earlier

- encrypted pdfs

Database Systems WiSe 2020/2021
Big Data Analytics Group
Matriculation number: 1234

Prof. Dr. Jens Dittrich
Final Exam
February 24th 2021

SAARLAND UNIVERSITY
COMPUTER SCIENCE

6 Performance Scaling (15 Bonus Points)

Consider a university that wants to perform online open book exams for their students. Each exam is slightly personalized, i.e. slightly varied using Python scripts parameterized among other information through the matriculation number (e.g. 1234) of the student. The output of those scripts is embedded in a LaTeX document. Compiling a single exam for one student (from LaTeX to pdf and including the costs for executing the Python scripts) takes 2 seconds. The resulting pdf has a size of $\frac{1}{3}$ MB. The I/O and network bandwidth of the web server is limited by 10Gbits/second.

Each exam has a start time t (that might be shared with other exams) and solutions have to be handed in by $t + 150$ minutes. At any point in time t, multiple exams may start with a total of N_t students for those exams, where N_t may be in the thousands.

At the start of the exam, students download their personal version from a webserver (e.g. http://a.b.de) by entering their matriculation number (e.g. 1234) which triggers the LaTeX compilation process and the following download.

a) Identify possible performance issues that may occur in this scenario. (3 Points)

Lessons Learned

Pros

- effort ok, can then be reused multiple times anyways
- easy to integrate into existing exam-documents
- works for several types of questions:
  at least algorithmic questions, multiple choice
- scales
- actually would allows us to identify sinks in plagiarism (with high likelihood) …
  [obviously we did not follow this path]

Cons

- somewhat more correction effort:
  need to correct each exam with personal solution
- be careful with testing: LaTeX swallows stderr
  (easy fix, testcases for all possible parameters, i.e. the current seed and all mat#s)

Future Work

explore extension to other types of questions