CSCE604135 • Temu-Balik Informasi
Semester Genap 2024/2025
Fakultas Ilmu Komputer, Universitas Indonesia

# Tugas Pemrograman 2: *Boolean Retrieval*

**Tenggat Waktu: Senin, 17 Maret 2025, 23.55 WIB**

## Ketentuan:

1. Anda diberikan beberapa *file* kode Python yang berisi *template* kode.
2. Kumpulkan **semua program** (*.py*) dan **laporan** (*.pdf*) jika ada, yang sudah dibuat dalam format *.zip* dengan penamaan **TP1_NPM.zip** melalui submisi SCeLe.
   Contoh penamaan file: TP1_2206123456.zip
3. Kumpulkan *file zip* tersebut pada submisi yang telah disediakan di SCeLe sebelum **Senin, 17 Maret 2025, 23.55 WIB**. Keterlambatan pengumpulan akan dikenakan penalti.
4. Tugas ini dirancang sebagai tugas mandiri. **Plagiarisme tidak diperkenankan dalam bentuk apapun**. Adapun kolaborasi berupa diskusi (tanpa menyalin maupun mengambil jawaban orang lain) dan memanfaatkan informasi dari literatur manapun masih diperbolehkan. **Pastikan** untuk mencantumkan nama kolaborator dan referensi literatur.
5. Anda boleh berkonsultasi terkait tugas ini asisten dosen berikut. Asisten dosen diperbolehkan membantu Anda dengan memberikan petunjuk.
   a. Inaya Rahmanisa
      Email: inaya.rahmanisa@gmail.com
      Discord: floringham
   b. Muhammad Falensi Azmi
      Email: falensiazmi@gmail.com
      Discord: frinzthecoder

# Petunjuk Pengerjaan Tugas

Pada tugas ini, Anda akan mengimplementasikan *inverted index* pada dataset yang diberikan dengan skema *blocked sort-based indexing* (BSBI). Sebelum melakukan *indexing*, ingat bahwa Anda wajib untuk melakukan *pre-processing*, seperti *tokenization*, *stemming*, dan *stopwords removal*. Dalam melakukan *indexing*, Anda wajib melakukan *compression* dengan menggunakan dua metode berbeda, **VB Encoding** (seperti yang dipelajari di kelas) dan metode kompresi lain, yaitu **Simple-8b** (penjelasan dapat dilihat di bawah). Setelah melakukan *indexing*, program Anda diharapkan dapat dijalankan untuk melakukan *boolean retrieval* berdasarkan *query* yang diberikan.

**Penjelasan Metode Kompresi Simple-8b**

Simple-8b adalah salah satu algoritma kompresi bit-level dari *simple* family yang dibuat oleh (Anh & Moffat, 2010). Metode ini merupakan pengembangan dari Simple-9, dengan fokus pada efisiensi dalam penyimpanan dan pemrosesan.

Karakteristik Simple-8b:

- Simple-8b terdiri dari **64 bits**
- **4 bit pertama adalah selector**
- **60 bit selanjutnya merepresentasikan angka yang dikompresi**.

Berikut ini *selector table* untuk Simple-8b.

| selector value | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| integers coded | 240 | 120 | 60 | 30 | 20 | 15 | 12 | 10 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| bits per integer | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 10 | 12 | 15 | 20 | 30 | 60 |

Contoh alur kompresi Simple-8b:

Misalnya kita ingin mengompresi *gap list* berikut.

```
[16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]
```

Panjang list: **12**

Nilai maksimal: **27**

Jumlah bit maksimal (*bits per integer*): **5 bit**. Karena nilai maksimalnya, 27, bisa disimpan dalam 5 bit.

Berdasarkan *selector table*, untuk *bits per integer* 5 bit, selector 6 (selector di mana selector value = 6) bisa memuat 12 integers, sehingga **selector 6 dipilih.**

Jadi, hasil encoding Simple-8b menjadi:

- Selector part: angka 6 dalam 4 selector bits

  `0110`

- Number part (right to left): 12 angka pertama di list dalam 5 bits

  `11011 11010 11001 11000 10111 10110 10101 10100 10011 10010 10001 10000`

Hasil encoding Simple-8b:

`11011 11010 11001 11000 10111 10110 10101 10100 10011 10010 10001 10000 0110`

Proses encoding ini terus dilakukan untuk seluruh gap dalam list. Khusus untuk metode Simple-8b, Anda dibebaskan untuk bereksplorasi dan mengimplementasikan berbagai konvensi algoritma Simple-8b selama memenuhi definisi.

**Boolean Retrieval**

Sistem *boolean retrieval* yang Anda buat diharapkan dapat mendukung pencarian dengan operasi AND, OR, dan DIFF (silakan merujuk pada *slides* untuk implementasi algoritma ini pada *iterables* yang sudah *sorted*). Penggunaan tanda kurung juga harus diimplementasikan untuk menunjukkan *precedence* dari operasi—yang mana yang perlu diselesaikan terlebih dahulu. Akan tetapi, ingat bahwa *precedence* antara AND, OR, dan DIFF setara karena ketiganya merupakan *set operators*.

Sebagai referensi, Anda dapat menggunakan ekspresi *postfix* untuk mengevaluasi *query* yang diberikan. Secara algoritma, evaluasi dalam ekspresi *postfix* akan jauh lebih mudah dibandingkan dengan menggunakan ekspresi *infix*. Anda dapat melakukan eksplorasi terkait algoritma Shunting-Yard untuk mengonversi ekspresi *infix* ke *postfix*. Sebagai contoh, ekspresi *postfix* dari operasi `2 + 3` adalah `2 3 +`. Contoh implementasi yang lebih jelas untuk tugas pemrograman ini dapat Anda lihat di *template* kode yang diberikan.

Contoh *query* **valid** yang dapat diberikan oleh pengguna adalah sebagai berikut.

- `term1 AND term2`
- `term1 AND (term2 OR term3)`
- `term1 AND (term2 OR (term3 DIFF term4))`
- `term1 AND term2 DIFF (term3 OR (term4 AND term5))`
- `term1 OR ((term2 AND term3) DIFF (term4 OR term5))`

Contoh *query* **tidak valid** adalah sebagai berikut. Program Anda tidak akan diuji berdasarkan ini.

- `term1 AND`
- `(term1 AND) term2`
- `term1 AND ((term2 OR term3)`

Jangan lupa untuk melakukan *pre-processing* pada *query* juga. Akan tetapi, Anda tidak perlu melakukan *stopwords removal* pada tahap ini, dengan asumsi tidak ada *stopwords* yang akan dimasukkan pada *query*. Terdapat suatu *method* di class `QueryParser` untuk melakukan validasi apakah suatu *query* mengandung *stopwords* atau tidak. Panggil *method* ini untuk melakukan validasi *query*, lalu kembalikan *list* kosong dan pesan bahwa terdapat *stopwords* pada *query*, jika memang terdapat *stopwords* pada *query*. Alasan mengapa *stopwords removal* pada *query* tidak diperlukan dikarenakan untuk mencegah suatu ekspresi hanya mempunyai satu *operand*, seperti jika `term1` dianggap sebagai *stopword*, maka setelah *stopwords removal*, ekspresi `term1 AND term2` hanya akan menjadi `AND term2`. Ini tentu akan menimbulkan isu pada evaluasi.

## Urutan Pengerjaan Tugas

1. `util.py`, *file* ini berisi berbagai *classes* dan *methods* pembantu untuk *files* lainnya.
2. `compression.py`, *file* ini berisi berbagai algoritma kompresi yang akan digunakan untuk *indexing*. *File* ini dapat dikerjakan secara paralel dengan `util.py`, karena tidak ada *dependencies* antara keduanya.
3. `index.py`, *file* ini berisi abstraksi *class inverted index*. Pastikan Anda telah menyelesaikan `compression.py` sebelum mengerjakan *file* ini.
4. `bsbi.py`, *file* ini berfungsi untuk melakukan *indexing*. Pastikan semua *files* yang telah disebutkan sebelumnya sudah dikerjakan sebelum mengerjakan *file* ini.
5. `search.py`, *file* ini berfungsi untuk melakukan *boolean retrieval*.

## Dataset

Anda akan bekerja dengan sampel dataset dari kumpulan abstrak dokumen di arXiv dalam bahasa Inggris. Total dokumen yang ada sebanyak kurang lebih 250.000 dokumen yang tersebar di 25 *folder*, di mana masing-masing *folder* berisi kurang lebih 10.000 dokumen. Tiap *folder* akan merepresentasikan satu *intermediate index*.

## Bonus

Anda akan diberikan bonus 10 poin jika Anda mengimplementasikan satu algoritma kompresi tambahan **selain VB Encoding** dan **Simple-8b** setelah selesai mengerjakan tugas wajib. Setelah itu, buatlah suatu laporan yang berisi waktu *indexing* dan ukuran *file indices* yang terbentuk untuk masing-masing algoritma kompresi. Bandingkan kedua aspek tersebut untuk algoritma-algoritma kompresi yang Anda telah buat dan berikan justifikasi mengapa hal tersebut bisa terjadi. Lampirkan laporan dalam bentuk `pdf`.

## Template Kode dan Dataset

- Template kode: tp2_template.zip
- Dataset: arxiv_collections.zip

## Deliverables

Kompres semua *files* dan *folders* berikut ini ke dalam suatu zip *file* dengan format penamaan seperti yang disebutkan pada bagian sampul dokumen soal ini. Anda tidak perlu mengumpulkan *folder* `arxiv_collections`, cukup dua (atau tiga, jika mengerjakan bonus) *folder index* yang dibuat saja.

- `bsbi.py`
- `compression.py`
- `index.py`
- `search.py`
- `util.py`
- `/index_vb`
- `/index_simple8b`

Jika *folder index* yang dihasilkan terlalu besar sehingga tidak bisa di-*upload* di SCeLE, silakan meng-*upload folder* tersebut di *cloud storage* seperti Google Drive terlebih dahulu, lalu sertakan URL-nya di dalam zip *file deliverables*. Pastikan URL tersebut dapat diakses.

**Catatan Revisi**:

-

## Rubrik Penilaian

| Komponen | Proporsi |
|---|---|
| Implementasi `bsbi.py` | 40% |
| Implementasi `index.py` | 20% |
| Implementasi `util.py` | 20% |
| Implementasi `compression.py` | 20% |
| Implementasi algoritma kompresi tambahan (Bonus) | 5% |
| Laporan (Bonus) | 5% |

**Selamat mengerjakan!**