W1: RAM vs EM Algorithms

- **Due:** Sunday Feb 18
- **Summary:** 3 questions and one challenge question, for a total of 12 points.
- **Submission:** https://autolab.cse.buffalo.edu/courses/cse410-s24/assessments/W1-Binary

Submission

Only PDF-formatted files will be accepted by autolab.

- You may write out your answers by hand and scan them; Numerous apps exist for phones to 'scan' in written documents as a PDF.
- You may typeset your answers in LaTeX, Typst, or a similar tool.

Note that the instructor must be able to read your answer. Submissions that are unintelligible will receive no points.

Binary Search

Recall the basic binary search algorithm.

```
fun binary_search(target: u32, data: Vec<u32>) -> usize
  { return binary_search(target, data, 0, data.len()); }
fun binary_search(target: u32, data: Vec<u32>, start: usize, end: usize) -> usize
{
  if(start >= end-1) { return start }

  let mid = (start - end) / 2 + start;

  if( data[mid] == target )     { return mid; }
  else if( data[mid] < target ){ return binary_search(target, data, mid, end); }
  else                          { return binary_search(target, data, start, mid); }
}
```

Question 1: Binary Search in RAM [4pt]

Let $T(N)$ be the runtime of the binary search algorithm given above where $N =$ `data.len()`:

1. Set up the recurrence relation for $T(N)$ (i.e., define T(N) by cases, in terms of itself).
2. Set up the base and recursive cases for a proof by induction that $T(N) = O(\log_2(N))$
3. Complete the proof by recursion that $T(N) = O(\log_2(N))$

Question 2: Binary Search in EM [4pt]

Assume that:

- `data` is initially stored in external memory (i.e., on disk), as it is in P1.
- Each disk page stores P `u32` values.

Let $I(N)$ be the number of page reads (i.e., the IO Complexity) of the binary search algorithm given above, where N is defined as above.

1. Set up the recurrence relation for $I(N)$.
2. Set up the base and recursive cases for a proof by induction that $I(N) = O(\log_2(N))$
3. Complete the proof by recursion that $I(N) = O(\log_2(N))$

ISAM Index

Remember the ISAM index structure we discussed in class? For $N =$ `data.len()` records, and P `u32` values per page, the index is a tree built as follows:

- The 1st level contains P `u32` values on 1 page, taken at uniform intervals from `data`
- The 2nd level contains P^2 `u32` values on P pages taken at uniform intervals from `data`
- …

- The ith level contains P^i `u32` values on P^{i-1} pages, taken at uniform intervals from `data`
- …
- The last level contains all of `data`.

To find a value (let's call this `isam_find`):

1. We do a binary search on the 1st level page. Say the value is between the i and $i+1$th elements on the 1st level page (or simply greater than the ith element if $i = P - 1$).
2. We do a binary search on the ith page of the 2nd level. Say the value is between the j and $j+1$th elements on the ith 2nd level page (or greater than the jth element if $j = P - 1$)
3. Repeat the process, descending levels until we identify the specific page of data.

If you prefer code, this algorithm is summarized as follows:

```
fun isam_find(target: u32, data: ISAM) -> Option<usize>
  { isam_find(target, data, 0, 0); }
fun isam_find(target: u32, data: ISAM, level: u32, page: usize) -> Option<usize>
{
  let current_page: Vec<u32> = data.get_page(level, page);
  let position = binary_search(target, current_page);
  if(level >= data.depth())
  {
    return page * data.page_size() + position;
  }
  else
  {
    return isam_find(target, data, level+1, page * data.page_size() + position)
  }
}
```

Question 3: ISAM Index in EM

Assume that you have an ISAM index structure, as defined above, stored on disk. Let $I_{\text{ISAM}}(N, P)$ be the number of page reads (i.e., the IO Complexity) of the `isam_find` algorithm defined above.

1. Draw the recurrence diagram[1]
2. Use the recurrence diagram to make a guess about the asymptotic bound on $I_{\text{ISAM}}(N, P)$.
3. Set up the recurrence relation for $I_{\text{ISAM}}(N, P)$ given the bound you guessed above.
4. Complete the proof by recursion for the bound you guessed on $I_{\text{ISAM}}(N, P)$.

Challenge Question [no points]

Assume you have an on-disk array of records **in sorted order**. What is the IO complexity of building an ISAM index, and what is an algorithm that achieves this bound.

[1] e.g., see https://cse.buffalo.edu/courses/cse250/2023-fa/slides/lec12-c.pdf, slide 5