ICCS200: Assignment homework-2

Kriangsak Thuiprakhon
kriangsak.thi@student.mahidol.edu

1: LSH

(i) From the given condition, notice that there are two cases to consider:

- when $|x - y| \geq w$
  In this case,
  $$1 - \frac{1}{w}|x - y| \leq 0 \rightarrow \mathbf{Pr}[f(x) = f(y)] = 0$$
- when $|x - y| < w$
  If this is the case, then:
  $$\mathbf{Pr}[f(x) = f(y)] = max(0, 1 - \frac{1}{w}|x - y|)$$
  Now we need to find that what values of $s \in [0, w]$ would the following statement holds
  $$f(x) = \lfloor \frac{x + s}{w} \rfloor = \lfloor \frac{y + s}{w} \rfloor = f(y)$$
  From this, we can also make an observation that the above holds if and only if
  $$s \notin_R [wx, w|x - y|]$$
  Hence,
  $$\begin{aligned} Pr\{s \notin_R [wx, w|x - y|]\} &= 1 - Pr\{s \in_R [wx, w|x - y|]\} \\ &= 1 - \frac{|x - y|}{w} \end{aligned}$$

Now, if we sum up the two cases:

$$Pr[f(x) = f(y)] = max(0, 1 - \frac{1}{w}|x - y|)$$

2: Dual Binary Search and Dual Merge Sort

(i) In the given handout, KTHSMALLEST function is written so that each time, the algorithm halves the array into two arrays of length $n/2$
From this we can see that the span shrinks by a factor of two each time it recurses, then the work and span will be at most log|A| + log|B|

(ii) New span bound with use of KTH-FUNCTION

$$
\begin{aligned}
&\texttt{mergeFway}(A, B, R, f) \ = \\
&\quad \%\ \textit{Same base cases} \\
&\quad \texttt{otherwise} \Rightarrow \\
&\qquad l = (|R| - 1)/f(|R|) + 1; \\
&\qquad \textbf{parfor}\ i\ \texttt{in}\ [0 : f(|R|)] \\
&\qquad\quad s = \min(i \times l, |R|); \\
&\qquad\quad e = \min((i + 1) \times l, |R|); \\
&\qquad\quad (s_a, s_b) \ = \ \texttt{kth}(A, B, s); \\
&\qquad\quad (e_a, e_b) \ = \ \texttt{kth}(A, B, e); \\
&\qquad\quad \texttt{mergeFway}(A[s_a : e_a], B[s_b : e_b], R[s : e]); \\
&\qquad \textbf{return};
\end{aligned}
$$

Note that the code is taken from the given handout From this we can derive the span of merge for two sorted sequences with the adoption of KTH-FUNCTION

(iii)

- ***Work*** with $f(n) = \sqrt{n}$
$$W(n) = \sqrt{n}W(\sqrt{n}) + O(\sqrt{n}logn)$$
- ***New Span*** with $f(n) = \sqrt{n}$
$$S(n) = S(\sqrt{n}) + O(logn)$$

(v) Upgraded Merge $work$ and $span$ bounds

- ***WorkBound*** with $f(n) = \sqrt{n}$
$$W(n) = \sqrt{n}W(\sqrt{n}) + O(\sqrt{n}logn)$$
which solves to O(n)
- ***New Span Bound*** with $f(n) = \sqrt{n}$
$$S(n) = S(\sqrt{n}) + O(logn)$$
which solves to O(logn)

(vi) Upgraded Merge Sort $work$ and $span$ bounds
giving $O(n)$ work and $O(log^2 n)$ span.

3: Quick Sort Span

Claim 0.1. *The span of partitioning an array is O(logn)*

Also, from our last assignment and what discussed in class, it has been shown that the depth (span) of a Treap is O(logn) ***w.h.p***. Hence ,

$$S(n) = \underbrace{O(logn)}_{thespanofaTreap} \times \underbrace{O(logn)}_{*} = O(log^2 n)$$

* is the span of partitioning an array when recursing on a treap.

4: String Comparison

To do string comparison we will adopt the use of MAP and SCAN functions

- Let array A be a mapped of strings X, Y with the corresponding COMPARE(X,Y) function, do this with pfor
- apply SCAN(⊕, 0, A) where:

$$\oplus := \begin{cases} A[i+1] & if A[i] = 0 \text{ and return } A[i+1] \\ A[i] & otherwise \end{cases}$$

```
def CP_par(X,Y):
    A = an array of length min(X,Y)
    pfor i in range(min(X,Y))
    A[i] = map(*(X,Y): if x<y => -1, x=y => 0 else 1)
    #then apply scan on collection A
    if A[i] =0, look up for A[i+1]
    #do this until we find the first A[i+1] !=0 then
    return A[i+1]
```

From the above pseudocode, we can see that the algorithm will do:

$$W(n) = min(m, n)$$

as we can only compare up to the smallest length of the two strings

$$S(n) = logmin(m, n)$$

because we will do scan on the array A which is of size min(m,n) 080163

5: Parallel Closest Pair

To analyze the span of Closest pair: we can do divide and conquer and then throw the two n/2 pieces to run recursively in parallel. Let's do try to write a peudocode:

- Compute separation line L such that half the points are on each side.

- $(d_1, d2) \leftarrow$ Closest Pair in the left half || Closest Pair in the right half
- $d \leftarrow min(d1, d2)$
- Delete all points further than d from $L \rightarrow O(1)$done by pfor
- Sort points in y-order $\rightarrow O(log^2 n)$ by quick sort
- Scan points in y-order and compute distance between each point and next constant number of neighbors, and update d accordingly $\rightarrow O(1)$ done by pfor
- return d

recurrence:

$$S(n) = S(n/2) + O(1) + O(log^2 n) + O(1) \rightarrow O(log^3 n)$$