Supplementary Material for
“Improving Read Mapping Using Additional Prefix Grams”

S1 Candidate Generation Algorithm

Algorithm 1: GenerateCandidates($L, k, \Delta_{gap}$)

input: $L$ is a set of inverted lists,
$k$ is a distance threshold,
$\Delta_{gap}$ is an allowed gap between two matched $q$-gram (it is set to
0 for hamming distance and set to $k$ for edit distance).

output: A set of candidate locations $C$.

1 sort inverted list in $L$ by their length;
2 foreach element $loc \in L[1]$ do $C$.push_back($loc - L[1].gramloc, 1$);
3 for $p \leftarrow 2$ to $k + 2$ do
   $C_{new} \leftarrow$ empty set of $(location, count)$ pairs;
   $i \leftarrow 1$;
   $j \leftarrow 1$;
   $carry \leftarrow 0$; /* deferred increment of count
/* assume each list contains max value at the end */
8 while $i \leq C.size()$ or $j \leq L[p].size()$ do
   candidate_loc $\leftarrow L[p][j] - L[p].gramloc$; // normalize $L[p][j]$ to find a candidate location
9   if $C[i].location - candidate_loc > \Delta_{gap}$ then
10      $C_{new}$.push_back($candidate_loc, 1 + carry$);
11      $j \leftarrow j + 1$;
12      $carry \leftarrow 0$;
13   else if $C[i].location - candidate_loc < \Delta_{gap}$ then
14      $C_{new}$.push_back($C[i].location, C[i].count + carry$);
15      $i \leftarrow i + 1$;
16      $carry \leftarrow 0$;
17   else
18      if $C[i].location = candidate_loc$ then
19         $C_{new}$.push_back($C[i].location, C[i].count + 1$);
20         $i \leftarrow i + 1$;
21         $j \leftarrow j + 1$;
22         $carry \leftarrow 0$;
23      else if $C[i].location > candidate_loc$ then
24         $C_{new}$.push_back($candidate_loc, 2$);
25         $j \leftarrow j + 1$;
26         $carry \leftarrow 1$; // set deferred increment of $C[i].count$
27      else
28         $C_{new}$.push_back($C[i].location, C[i].count + 1$);
29         $i \leftarrow i + 1$;
30         $carry \leftarrow 1$; // set deferred increment of the count of candidate_loc
31      $C \leftarrow C_{new}$;
33 remove every element from $C$ whose $count$ is less than 2;
34 Return $C$;
S2 Additional Experiments

**Effect of the gram length**: we ran an experiment to figure out the effect of the gram length, $q$, on the performance. We also compared the $k+1$ prefix gram technique with the $k+2$ prefix gram technique, where $k$ is an edit distance threshold. In the experiment, we mapped 500k 100 bp single-end reads against HG18 using 16 threads for an edit distance threshold 5. Figure S1 shows the results. In $k+1$ prefix gram technique, the performance of read mapping improved until $q = 13$. However, the performance degraded when $q = 14$. The $k+2$ prefix gram technique exhibited the best performance when $q = 11$. If we use large $q$, we reduce sizes of inverted lists and as a result we generate fewer candidates. As we increase $q$ size, however, the number of different grams grows sharply and we need much time to look up an inverted index to find an inverted list of each gram. Increasing $q$ size also affects selecting good grams.

![Figure S1: Performance results according to different gram lengths](image1)

**Effect of the number of additional prefix grams**: Figure S2 shows mapping times according to the number of additional prefix $q$-grams. For the experiment, we mapped 500k 100 bp single-end reads against HG18 using 16 threads for an edit distance threshold 5. To select more than one additional $q$-grams, we simply used the same heuristic described in the paper. As depicted in the figure, only the first additional prefix gram brought a significant improvement. The second and third prefix resulted in marginal improvements only.

![Figure S2: Performance results according to different additional prefix grams](image2)
Experiments with hamming distance constraints: since Hobbes2 runs exactly the same as Hobbes for short reads, we conducted experiments only on 76bp and 100bp reads of HG18 using a single thread for Hamming distance constraints. 76bp reads taken from DDBJ repository with entry DRX000360. Table S1 shows the fraction of reads with at least one mapping, the mapping time, and the peak memory consumption. We ran tools using a single thread in the experiments. Hobbes2 was slightly slower than Hobbes with 32 bit vectors but it was faster than Hobbes with 16 bit vectors. Hobbes2 was 3 to 6 times faster than Masai, GEM, and RazerS3. In terms of memory consumption, Hobbes2 required only 14.8 GB, which was about a half of the memory footprint of Hobbes with 32 bit vectors. GEM lost only a small amount of mappings for 76bp reads, but lost many mappings for 100bp reads. RazerS3 did not correctly map both single end and paired end reads.

<table>
<thead>
<tr>
<th>mapper</th>
<th>single end, 76bp, 3 errors</th>
<th>single end, 100bp, 5 errors</th>
<th>paired end, 100bp, 5 errors</th>
<th>peak memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>read mapped (%)</td>
<td>total mappings (million)</td>
<td>mapping time (min:sec)</td>
<td>read mapped (%)</td>
</tr>
<tr>
<td>Hobbes2</td>
<td>91.43</td>
<td>73.37</td>
<td>03:35</td>
<td>90.10</td>
</tr>
<tr>
<td>Hobbes-16bit</td>
<td>91.43</td>
<td>73.37</td>
<td>04:29</td>
<td>90.10</td>
</tr>
<tr>
<td>Hobbes-32bit</td>
<td>91.43</td>
<td>73.37</td>
<td>02:49</td>
<td>90.10</td>
</tr>
<tr>
<td>Masai</td>
<td>91.43</td>
<td>73.37</td>
<td>12:50</td>
<td>90.10</td>
</tr>
<tr>
<td>GEM</td>
<td>91.43</td>
<td>73.37</td>
<td>19:58</td>
<td>85.23</td>
</tr>
<tr>
<td>RazerS3</td>
<td>67.57</td>
<td>4.33</td>
<td>13:54</td>
<td>64.67</td>
</tr>
</tbody>
</table>

Experiments in m mapping mode: Hobbes2 supports m mapping mode, which reports only those reads such that the number of total mapping locations is less than or equal to a given threshold m. We ran experiments to evaluate Hobbes2 in m mapping mode. We mapped 500k 100 bp single-end reads against HG18 using 16 threads for an edit distance threshold 5. Table S2 shows the results. If m is very large, the mapping time is similar to that of all mapping mode (e.g. m = 10000 in Table S2). However, for a reasonable m value (up to 1000 in Table S2), m mapping mode is 1.5 to 2 times faster than all mapping mode while losing only a small set of reads.

<table>
<thead>
<tr>
<th>threshold (m)</th>
<th>mapped reads (%)</th>
<th>mapping time (min:sec)</th>
<th>peak memory (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>82.021</td>
<td>2:44</td>
<td>14.7G</td>
</tr>
<tr>
<td>10</td>
<td>87.222</td>
<td>2:57</td>
<td>14.7G</td>
</tr>
<tr>
<td>100</td>
<td>88.993</td>
<td>3:16</td>
<td>14.7G</td>
</tr>
<tr>
<td>1000</td>
<td>89.877</td>
<td>3:48</td>
<td>14.7G</td>
</tr>
<tr>
<td>10000</td>
<td>91.458</td>
<td>4:49</td>
<td>14.7G</td>
</tr>
<tr>
<td>all</td>
<td>91.476</td>
<td>5:17</td>
<td>14.7G</td>
</tr>
</tbody>
</table>

S3 Configuration of Each Tool

We configured read mappers used in our experiments as follows.

Hobbes2. We used Hobbes version 2.1. We built indices with 16 bit vectors for both edit distance and Hamming distance constraints. In order to use Hobbes2 as an all mapper we specified -a option. We used --hamming option for Hamming distance constraints and --indel option for edit distance constraints. For the paired end mapping, we set --min and --max parameters to 110 and 290 respectively. For multi-threaded execution, we used -p option. Note that Hobbes2 generates output in the SAM format with cigar strings by default.

Hobbes. Version 1.5 was used. We used 16 bit vector inverted index for edit distance constraints. Since Hobbes does not calculate cigar strings by default, we used --cigar option to produce output with cigar strings. Other configurations of Hobbes were the same as Hobbes2.
**RazerS3.** Version 3.2 was used. To use RazerS3 as an all mapper, we used set the parameter `-m` to 1,000,000. To measure the total number of mappings, we manually removed duplicated outputs. We do not include the time for duplication eliminations. To output results in the SAM format, we used the extension “.sam” in output files. For multi-threaded execution, we used `-tc` option. We set the parameters `-ll` and `-le` to 200 and 90 respectively for the paired end mapping. For Hamming distance constraints, we used `--no-gaps` option and set the parameter `-rr` to 100 to produce all possible mappings. We used the default value of the parameter `-rr` in edit distance case. For other options, we used the default values of RazerS3.

**Masai.** Version 0.7.1 was used. To use Masai as an all mapper, we used ‘-mm all’ option. We set `-ll` and `-le` parameters to 200 and 90 respectively for the paired end read mapping. In Hamming distance case, we specified `--no-gaps` option. We used the extension “.sam” in output files to output results in the SAM format. To produce results for paired end reads, we included the conversion time in the mapping time.

**BWA.** Version 0.7.4 was used. We used the `-N` option to configure BWA as an all mapper. To specify an edit distance 5, we set `-n`, `-o`, and `-e` parameters to 5, 5, and -1, respectively. For multi-threaded execution, we used `-t` option. We used the default values for other options.

**Bowtie2.** Version 2.1.0 was used. We used the `-a` option to configure Bowtie2 as an all mapper. To specify an edit distance 5, we used `--end-to-end` and `--ignore-quals` options and set `--mp`, `--np`, `--rdg`, `--rfg`, and `--score-min` parameters to [1,1], 1, [0,1], [0,1], and [L,0,−0.05], respectively. For multi-threaded execution, we used `-p` option. We used the default values for other options. We set `-min` and `-max` parameters to 210 and 390 respectively for paired end reads.

**GEM.** Version 1.376 was used. To specify an edit distance 5, we used ‘quality-format = ignore’ and set both `-m` and `-e` parameters to 5. To configure GEM as an all mapper, we set `-d` parameter to `all` and `-s` parameter to 1,000,000. We converted outputs of `gem-mapper` into SAM format outputs using `gem-2-sam`. We included the conversion time in the mapping time.