De La Salle University - Manila

Term 3, A.Y. 2023 - 2024

In partial fulfillment

of the course in

Parallel Computing (S11)

Integrating Project - Milestone #2

Submitted by:

Ambrosio, Carlos Felipe Q.

Arceta, Althea Zyrie Manuel

Mendoza, Antonio Gabriel Notario

Tan, Jose Tristan Turtoza

Submitted to:

Mr. Roger Luis Uy

July 7, 2024

The sequential version of the smith-waterman algorithm using the blosum62 matrix with an affine gap has been implemented as below:

```
C/C++
void mana(char* seq1, char* seq2) {
	int k = (int)strlen(seq1) + 1;
	int h = (int)strlen(seq2) + 1;
	int i,j;
	int mat[k][h];
	bool gap[k][h]; //keeps track of the gap
	for (i=0;i<k;i++) {
		for(j=0;j<h;j++) {
			if(i==0||j==0) {
				mat[i][j]=0;
				gap[i][j]=false;
			}
			else {
				int hgapsc = (!gap[i-1][j]) ? mat[i-1][j]-5 : mat[i-1][j]-1;
				int vgapsc = (!gap[i][j-1]) ? mat[i][j-1]-5 : mat[i][j-1]-1;
				int xscore = mat[i-1][j-1]+score(seq1[i-1],seq2[j-1]);
				if (hgapsc > vgapsc && hgapsc > xscore) {
					mat[i][j]=hgapsc;
					gap[i][j]=true;
				} else if (vgapsc > hgapsc && vgapsc > xscore) {
					mat[i][j]=vgapsc;
					gap[i][j]=true;
				} else {
					mat[i][j] = xscore;
					gap[i][j] = false;
				}
				if (mat[i][j]<0) {
					mat[i][j]=0;
					gap[i][j] = true;
				}
			}
		}
	}

	for (i=0;i<k;i++) {
		for (j=0;j<h;j++) {
			printf("%d ", mat[i][j]);
		}
		printf("\n");
	}
}
```

During the process of implementing the algorithm, several dependencies were found, namely the blosum62.txt file, the matrix keeping track of whether a gap would be an opening or an extension, and the matrix itself as each element, aside from the base cases, requires the scores of

the index one row, the index one column, and the index one row and one column behind it. The scoring based on the blosum62 matrix was implemented as below:

```C/C++
int place(char a) {
	switch(a) {
		case 'A': return 0;
		case 'R': return 1;
		case 'N': return 2;
		case 'D': return 3;
		case 'C': return 4;
		case 'Q': return 5;
		case 'E': return 6;
		case 'G': return 7;
		case 'H': return 8;
		case 'I': return 9;
		case 'L': return 10;
		case 'K': return 11;
		case 'M': return 12;
		case 'F': return 13;
		case 'P': return 14;
		case 'S': return 15;
		case 'T': return 16;
		case 'W': return 17;
		case 'Y': return 18;
		case 'V': return 19;
		case 'B': return 20;
		case 'J': return 21;
		case 'Z': return 22;
		case 'X': return 23;
		default: return 24;
	}
}

int score(char a, char b) {
	int dA,dB;
	dA = place(a);
	dB = place(b);
	return blosum62mat[dA][dB];
}
```

The place function gets the index of the given string, and the score function returns the score of those two strings provided by the blosum62 matrix. Looking ahead, the next steps for the project would be to implement the CUDA kernel using the wavefront algorithm, as well as the backtracking algorithm