情報－Ⅱ

学習指導要領（2）- 知・技 - ア
学習指導要領（3）- 知・技 - ア
学習内容（2）- ア メディアとコミュニケーション
学習内容（3）- ア コンピュータの仕組みと処理

(ア) 2進法による浮動小数点数の扱いについて述べた次の文章を読み、空欄に入るもっとも適切な数字をマークしなさい。

8ビットで浮動小数点数を表現することとし、2進法で、$(A_0A_1A_2A_3A_4A_5A_6A_7)_2$ と表す。これによって、次のような正の浮動小数点数を表すことにする。

$$(1.A_0A_1A_2A_3)_2 \times 2^{(A_4A_5A_6A_7)_2-7}$$

（注：実際に使われている形式では、0や無限大の表現のため、いくつかの2進法での表現が正の浮動小数点数を表さないが、ここでは考慮せず、8ビットの2進法での表現すべてがこの形式で使われるものとする。つまり、仮数部は、$A_0A_1A_2A_3$ の4ビットであり、隠れビットを用いた正規表現が用いられている。指数部は、$A_4A_5A_6A_7$ の4ビットであり、指数部のバイアスが7となっている。また符号はなく、正の数だけを表している。）

この形式で表現できる最大の数は、10進法で、(9)(10)(11)(12) である。

また最小値を10進法の分数で正確に表現すると、

$$\frac{(13)(14)(15)}{(16)(17)(18)(19)}$$

となる。

また、この表現を用いた場合、10101010という8ビットは、10進法で (20)(21)(22).(23)(24) となり、01100110は、(25)(26).(27)(28)(29)(30) となる。

また、6.25をこの浮動小数点形式に変換して、上記の8ビットでの表現形式で表現すると、(31)(32)(33)(34)(35)(36)(37)(38) となる。

学習指導要領（3）- 知・技 - イ
学習指導要領（3）- 思・判・表 - イ
学習指導要領（3）- 思・判・表 - ウ
学習内容（3）- イ アルゴリズムとプログラム
学習内容（3）- ウ モデル化とシミュレーション

(イ) 次の文章は、プログラムにおいて、表や辞書のようなデータ構造を効率よく処理するために用いられる方法について説明したものである。空欄に入るもっとも適切な数字を解答欄にマークしなさい。ただし、解答欄 (39)(40) から (51)(52) については、空欄にあてはまるもっとも適切な語を下の選択肢から1つ選び、その番号をマークしなさい。

英文のテキストデータがあり、各単語が何度使われているかを数えるプログラムを考える。テキスト

データの最初から単語を順番に読んで、その単語の使われている回数を記憶しているメモリ内の場所を決定し、その内容を1だけ増やす操作をテキストデータを読み終えるまで繰り返せばよい。

こう述べると簡単ではあるが、ここで問題となるのは、「使われている回数を記憶しているメモリ内の場所を決定し」の部分である。

どのような単語が使われているかわかっている場合は、あらかじめ各単語の使用回数を記憶する場所をメモリ内に割り当てておく方法が考えられる。使われる可能性のある単語が1,000,000個だったとすると、テキストデータから読み取った単語を最初から順番に比較していき、メモリ内のどの場所に回数を記憶しているかを決定すると、最小で1回の比較、最大で約 (39)(40) 回の比較が必要になる。また、比較の回数の期待値は、約 (41)(42) 回となる。2分探索法で調べられるように、順序をつけて並べるようなデータ構造を用いて、2分探索法で調べると、最大でも約 (43)(44) 回の比較ですませられる。

次に、あらかじめどのような単語が使われているかわからない場合を考える。最初に読み込んだ単語から順番に回数を記憶する場所を割り当てていく方法が考えられる。単純に読み込んだ単語を順番にメモリに配置していくようなデータ構造を用いることを考える。この場合、100,000種類の単語を読み込んだ後、まだ読み込んだことのない新しい単語が表れると、その単語を読み込んだことがないということを確認するのに、メモリを順番に比較して調べていくと、 (45)(46) 回の比較が必要になってしまう。ここでも、新しい単語が表れるたびに、2分探索法で調べられるように、データ構造を作っていけば、新しい単語を読み込んだ場合に必要な比較の回数を大幅に小さくできる。

効率よく単語を調べて回数を記憶するための方法として、2分探索法で調べられるデータ構造を使う方法のほかに、ハッシュ法と呼ばれる方法が知られている。

ハッシュ法は、対象となる単語から、ある方法でハッシュ値と呼ばれる数バイト程度の整数値を計算し、その整数が示す記憶場所をその単語の情報(この場合、現れた回数)を格納するのに使うというものである。単語から整数値を計算するだけなので、理想的な場合には比較は必要なくなるが、現実的には、後述のハッシュ値の衝突が起るのでそれを処理するための比較などを含む処理手順が必要になる。

ハッシュ値の計算はたとえば次のように行う。アルファベット26文字から構成される単語に含まれる各文字が数値に変換できるとし、10進法でAに65、Bに66、Cに67、そしてDからも順番に数値に変換でき、Xに88、Yに89、Zに90が割当てられているものとする。たとえば、ABCという単語に対して、65+66+67を計算して、198をハッシュ値とする。これを記憶場所を示す値として使う。すぐにわかるように、ABCと同じハッシュ値をもつ単語は、CBA、 (47)(48) などを始め、多く存在する。

これをハッシュ値の衝突と呼ぶ。また、 (49)(50) と (51)(52) も同じハッシュ値をもつ。衝突した場合には、登録されている元の単語と比較を行って、すでに現れたものかどうかを調べるなどの処理が必要になる。

したがって、できるだけハッシュ値の衝突が起らないほうが、全体の処理が少なくてすむ。このためには、ハッシュ値を計算する手順 (これをハッシュ関数と呼ぶ) に工夫が必要である。たとえば、A の 65 に B の 66 を単純に加えるのではなく、A の 2 進法表現を 1 ビットだけ左にシフト (つまり左に 1 ビットずらす。2 倍することに相当する) してから、加算することが考えられる。そうすると、AB のハッシュ値は、196 になる。同様に 3 文字目の C についても 2 文字目までの計算結果を 1 ビット左シフトしてから加算することにすると、ハッシュ値は、 (53)(54)(55)(56) になる。この方式を用いると、単純な加算によるハッシュ関数ではハッシュ値が ABC と同じになった CBA のハッシュ値は、 (57)(58)(59)(60) となり、異なることがわかる。

この方法だと、衝突する可能性は減るが、単語の文字数が増えるとハッシュ値が無制限に大きくなってしまい、プログラムで取り扱う場合に不便である。そこで、ハッシュ値をあるビット数以下に抑えるために、たとえば、計算を 64 ビット整数の範囲で行い、計算の途中で桁あふれした部分は無視する方法が考えられる。また、この場合でも、64 ビットのハッシュ値で指定される記憶場所を用意するのは現実的ではないので、64 ビットから 24 ビットや 16 ビットの値に計算し直す、あるいは最初から途中の計算も 24 ビットや 16 ビットになるような方法と使うなど、ハッシュ法については多くの研究が行われている。

【(39)(40)〜(51)(52) の選択肢】

(11) 10	(12) 100	(13) 1000	(14) 10000	(15) 100000
(16) 1000000	(17) 20	(18) 200	(19) 2000	(20) 20000
(21) 200000	(22) 50	(23) 500	(24) 5000	(25) 50000
(26) 500000	(27) AAA	(28) BBB	(29) CCC	(30) DDD
(31) ABS	(32) ADD	(33) CMP	(34) DIV	(35) MUL