track 試験解答データから読み解く プログラミング言語選択の考察

新田章太 †，竹内郁雄 ‡

概要　プログラミング学習者がどの言語を学ぶべきかは重要な選択であるが，実用目的に応じた適切な言語選択を裏付ける客観的なデータがない．そこで，我々は自ら開発したオンラインプログラミング学習・試験プラットフォーム track に集積されている 16749 人の学生のプログラミング問題解答データから，問題の難易度やカテゴリ別にプログラミング言語毎の正答率を分析した．

試験の解答のために選択されたプログラミング言語別の得点データから，C++と python3 の得点が高いことがわかった．また，カテゴリ別の得点の比較では，C++が木とグラフ，再帰とダイナミックプログラミング，探索とソートの問題で他を圧倒する成績を出し，Python3 では数学関連の問題で他を圧倒する成績を出していることもわかった．

これらの分析から，言語の選択や学習について考慮すべきことが浮び上がってきた．

キーワード　track，プログラミング試験，プログラミング言語の選択

1. はじめに

IT 技術者の需要が高まりつつある昨今において，2020 年から開始されている全国の小学校でのプログラミング教育の必修化など，プログラミング教育への関心が高まっている．初等，中高等教育に限らず，プログラミングスクール，オンライン学習サービスなどプログラミング学習手段は多岐にわたる．また，多くの大学の授業におけるプログラミングの入門言語としては C や Java 言語が代表的であるが [1]，文部科学省が 2019 年に発表した学習指導要領においては Python が採択され，大きな話題を集めたのは記憶に新しい．

このような流れを受け，教育者はどの言語を教えるべきか，学習者はどの言語を学ぶかについて，それぞれの立場で選択をしなければならなくなってきている．

教育者の目線においてどのプログラミング言語を採択するかを考えるには．標準的な機能を備えており教育に適しているか，生産性や保守性が高いか，研究活動に利用しやすいか，社会でのニーズが高いか，など様々な選択基準が考えられるが，これらに一意の解はなく，教育者に委ねられている．

また，学習者の目線におけるプログラミング言語の選択においても，特定の分野で研究や仕事をする上で効率的で扱いやすい言語かどうか，社会で求められる応用に適した言語であるかどうか，を理解した上で学習に取り組むことが重要である．しかし，現状では学習者がどの言語を学ぶかは教育者の指示や学習環境に依存しており，実用目的に応じた適切な言語選択を裏付ける客観的なデータがないため，学習者自らが言語を選択するのに支障をきたしている．

その中で，我々は自ら開発したオンラインプログラミング学習・試験プラットフォーム track[2] により，膨大な数の就職試験受験者の分析が可能になり，様々なプログラミング言語の解答の中から，どの言語の成績が良かったかを可視化することができた．

本論文では，企業が学生の選考時に技術試験として利用している track の問題・解答データから，問題の難易度やカテゴリ別にプログラミング言語毎の正答率を分析する．これにより，学習者が実用目的に応じた適切な言語選択を行うために参照できる客観的なデータを抽出する．

2. 調査内容

track にはプログラミング問題及び受講者の解答データが保存されているが，企業の採用選考時に技術試験として利用された結果が全体の約 8 割を占める．本研究では，track に保存されている受験者のプログラミング問題の解答結果の中から，解答数が一定数以上集まっている問題を選別し，対象の問題の解答得点を言語毎に難易度別，カテゴリ別に分析した．

得点の傾向を言語別に比較することで，特定のカテゴリに対して効率的な言語，あるいは学習すべき言語はどれであるかを考えるのに有用と思われる客観的なデータを抽出する．

† 株式会社ギブリー

‡ 東京大学名誉教授，株式会社ギブリー技術顧問

2.1 調査対象

2019 年 5 月 25 日から 2020 年 3 月 23 日までの間に，分析対象となる track の問題の解答をしているデータ

ユニークな解答者数: 16749 人
解答数: 29243 個

を調査対象とする．文部科学省の学校基本調査 [3] によると，情報処理・通信技術者への大卒の就職者数は約 2 万 8 千人程度であることから，今回の調査対象は，情報処理・通信技術者として就職活動をしている受験者のデータとして十分な量である．

2.2 問題（チャレンジ）の選定

今回はプログラミング言語別の解答結果の比較を実施することが目的のため，受験者がプログラミング言語を選択して解答することが可能なコーディング形式（2.5 節参照）の問題に限定して解答データを取得した．また，数ある問題セットの中から，集計期間内に 100 解答以上あることを条件に問題を 48 問選別した．なお，track を現在利用している企業約 150 社のうちの約 20 %では，担当技術者が track の問題作成システムを用いて独自の問題を作成しているが，これらは今回の分析の対象外とした．

2.3 問題の難易度の定義

track の問題には，1（初級），2（中級），3（上級）の難易度が設定されている．それぞれの難易度別の定義は以下の通りである．

1 （初級）: 基本なプログラム構造（if 文や for 文など）の理解や簡単なデータ操作が求められる問題
2 （中級）: ソーティング，グラフ，ダイナミックプログラミング，キュー，二分探索などの一般的なアルゴリズム実装が求められる問題
3 （上級）: 一般的なアルゴリズム実装だけではなく，発想力，高度なアルゴリズムやデータ構造，重実装が必要になる問題

2.4 問題のカテゴリの定義

Gayle Laakmann McDowell 著の“Cracking the Coding Interview”[4] はプログラミング入社試験対策について書かれた定評のある本である．ここで提案されている分類で track の問題をタグづけし，どの言語での解答がどのタグにいい成績を収めたかを調べた．

BA (Basic concepts)
基本データ型（以下に別分類されたデータ構造を除く），条件分岐（if, case 等），ループ（for, while 等）．以下のものにはこれらが当然含まれているので，以下を含まないもののみ BA とする．

AS (Arrays and Strings)
添字操作を行うもの全般（ビット操作を除く）．ハッシュ表もここに入れる．

LL (Linked Lists)
連結リストを使うのがデータモデルとして適切な問題．配列で連結リストを表現している場合は AS と LL の両方のタグづけをする．

SQ (Stacks and Queues)
スタックやキューをデータ構造として使うことが想定されるもの．

TG (Trees and Graphs)
木構造やグラフ構造でモデル化されることが想定されるもの．配列でこれらが表現されている前提のものは AS タグもつける．ポインタを使って表現されている前提のものは LL タグもつける．

BI (Bit Manipulation)
ビット操作を行うことが想定されるもの．ビットテーブルの場合は AS タグもつける．

MA (Mathematics)
代数，幾何，順列・組合せ，確率・統計，数列，数値計算・誤差，FFT，乱数．衝突判定，物理学的計算を含む．

LG (Logic)
ブール代数，論理回路，論理パズル．

RD (Recursion and Dynamic Programming)
再帰関数，再帰手続きでモデル化することが想定されるもの．再帰関数のループ化を含む．再帰関数のメモ化は SS と両方のタグをつける．ダイナミックプログラミングの範疇に入るものを含む．

SY (System Design and Scalability)
Scalability の意識を持たせる問題にはこのタグをつける．例: ネットを通した API 呼び出し回数の最適化．

SS (Sorting and Searching)
ソーティングと広義の探索に関するものにはすべてこのタグをつける．例: 貪欲法，ゲーム木探索，最短経路探索，A*探索など．

AD (Advanced)
上級問題にはすべてこのタグがつく．このほか，上記に含まれないものすべて，例えば，コンカレンシー，並列プログラミング，Two Pointer algorithm, 高度なデータ構造など．上記に含まれるものでも，上級と判定されるもの（重複タグ可）．[参考: 上記の Cracking 本 [4] Chapter 16 Medium, Chapter 17 Hard, XI Advanced を参照]

なお，これらのカテゴリが受験者に明示されることはない．受験者は問題文を読んで，その問題にどのようなアルゴルズムを使うかべきか，自分で判断しないといけない．当然だが，必要なアルゴリズムのテンプレートなどは与えられない．

2.5 問題の具体例

ここで考察する問題は，与えられた要件を満たすプログラムを書く問題である．受講者は問題を開くと画面左側に問題文，右上にコーディングエディタ，右下にプログラムを実行した際のコンソールが表示されている（図 1）．受講者が解答に利用したいプログラミング言語を選択すると（企業によっては選択範囲が限定されることがある），言語毎の基本テンプレートが与えられた状態から解答を始める．テンプレートは，テストで与えられる標準入力をそのまま標準出力するプログラムとして与えられる．

問題には複数（10〜30 個が典型的）の満たさなければならない要件がテストケースとして指定されており，受講者はすべての入力ケースに対して正しい出力を返すプログラムを書くことが要求される．なお，テストケースには受験者には見えないものが含まれている．このような多数のテストケースを用意することにより，プログラミングの問題に対しても部分点が与えられることになる．

図 1 に具体例として難易度 1，カテゴリ BA の問題の主文を抜粋したものを示した．ただし，問題文の正確な情報は試験問題の漏洩になるので，ここでは問題の概要説明のみをスクリーンショットで紹介し，一部の具体的な指示などはマスクしてある．この問題は，現在の体重・身長から目標の BMI 値を達成するためには何 kg 減量する必要があるかを求める問題である．

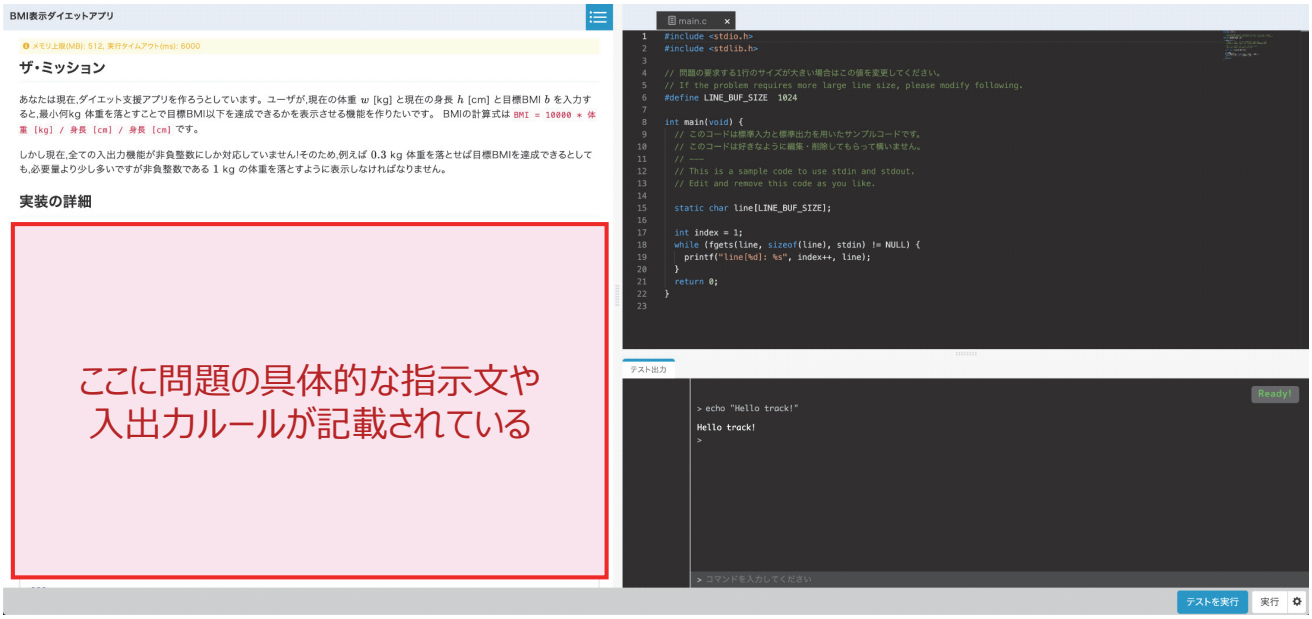

図 1. 問題とそれを解くためのウィンドウ

表 1. プログラミング別得点分布（解答数）

得点	C	C#	C++	Go	Java	JavaScript	Kotlin	Perl	PHP	Python2	Python3	Ruby	Scala	Swift
0	1534	326	587	55	1189	260	10	11	422	360	2218	555	11	16
1-9	138	45	93	3	114	24	7	2	44	30	247	49	3	4
10-19	135	40	103	7	119	28	3	4	53	42	482	67	1	4
20-29	116	32	104	13	113	54	2	1	48	32	427	75	3	3
30-39	126	43	87	6	124	48	4	0	42	41	322	75	2	2
40-49	155	66	139	9	176	41	4	2	68	26	369	64	3	2
50-59	78	28	89	3	89	21	1	0	48	27	352	32	1	0
60-69	73	19	90	8	74	23	0	0	62	29	298	27	0	1
70-79	97	25	122	23	117	30	2	1	62	25	497	42	3	4
80-89	57	30	158	10	67	49	1	8	65	41	578	102	1	2
90-99	200	73	291	16	223	39	1	5	94	62	1051	120	10	0
100	521	361	1851	118	852	352	9	17	368	266	5499	583	38	17

3. 分析

3.1 プログラミング言語別全体分析

まず，表 1 にすべての問題のプログラミング言語別の得点の分布を，ゼロ点，満点（100 点），その中間を 10 点刻みでまとめた結果を示す（最初は 1 点〜9 点）．見てすぐわかるように 0 点と満点に分布が偏った U 字型分布をしている．

コーディング形式の問題では，図 2 に示した山型分布になるクイズ形式（選択・穴埋め問題の総称）とは異なり，ゼロ点または満点に分布が偏る．図 3 は表 1 に掲載したすべてのコーディング形式の問題の得点分布を表したものである．通常これは識別性が高いことを意味するが，ある程度までできているという部分点評価に対する需要は採用企業側には根強い．簡単なものを含めて多数のテストケースを用意したのはこのためである．

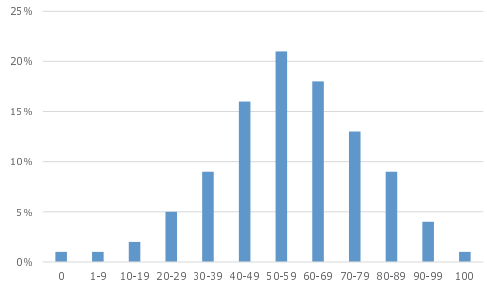

図 2. クイズ形式の山型得点分布

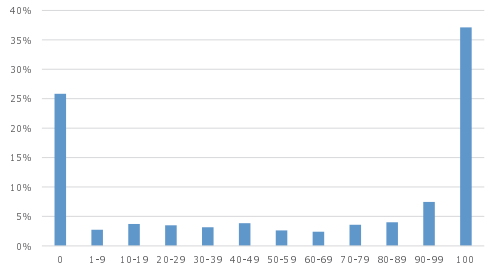

図 3. コーディング形式の U 字型得点分布

表 2 は 500 受験以上のプログラミング言語別の受験数と平均点を，難易度毎にまとめたものである．また，得点がゼロ点と満点に偏るため，表には示さないが，標準偏差はほぼどの言語でも 40〜45 と大きい．

U 字型分布ではあるが，表 2 から，プログラミング言語別の平均点を比較した．平均点の近いプログラミング言語をグループ化すると，言語別の成績は難易度に関わらず，表 3 のように 3 つのグループに分かれることが読み取れる．

解答の得点は，受験者のプログラミング能力が高い，ライブラリの充実などでプログラミング言語が使いやすい，大量の計算に対応できる実行速度を持つといった複合的要因に影響されると考えられるため，このグループの差異は「人＋言語」の能力の差と捉えられる．なお，部分点を考慮せずに，ゼロ点と満点の比率を比べると，「人＋言語」の差が鮮明に浮かび上がる（図 4）．この極端な U 字型分布の平均はゼロ点・満点比を表す良い近似になっている．

表 2. プログラミング言語・難易度別受験数・平均点

	難易度1		難易度2		難易度3		全体	
言語	受験数	平均点	受験数	平均点	受験数	平均点	受験数	平均点
C++	1622	70.98	1532	68.66	560	68.32	3714	69.62
Python3	5424	73.29	5494	60.44	1422	60.80	12340	66.13
JavaScript	531	58.54	363	50.49	75	47.23	969	54.65
Ruby	935	54.89	675	48.52	181	47.50	1791	51.74
C#	717	53.95	296	46.90	75	43.64	1088	51.32
PHP	685	52.83	614	47.02	77	45.26	1376	49.81
Python2	433	48.87	458	41.24	90	54.91	981	45.86
Java	1721	48.69	1181	41.41	355	39.51	3257	45.05
C	1871	39.75	1096	24.35	263	26.71	3230	33.46

表 3. 得点別に分けた 3 つの言語のグループ

C++, Python3	> 65 点
JavaScript, Ruby, C#, PHP, Python2, Java	≈ 50 点
C	< 35 点

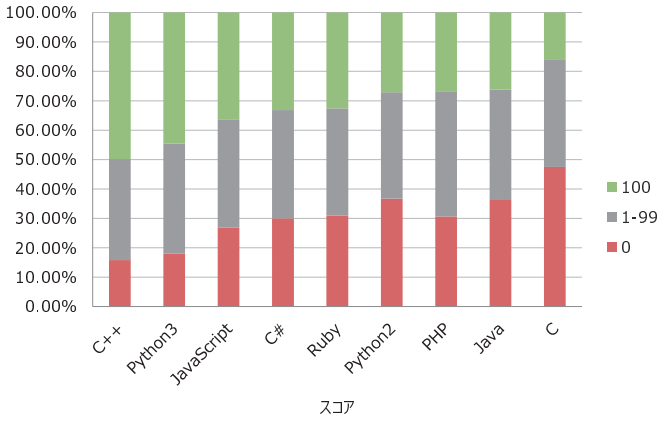

図 4. 言語別のゼロ点・満点比率

このような結果をより深く分析するため，成績優秀者の多い C++，Python3 で良い成績を出した人に対してアンケートを実施した．これについては第 4 章で述べる．

3.2 難易度別分析

次に，問題の難易度別の得点分布を分析した（表 4）．

表 4 から，難易度 1，2，3 それぞれの部分点取得比率は 23.93%，46.39%，60.89%となっている．このことから，難易度が高くなるにつれて中間点を取得する受講者の比率が高くなっていることがわかる．これは問題の設計上，難易度が低い問題は基本的なプログラムの実装のみを評価するテストで構成されるため，All or nothing になりやすく，一方で難易度が高い問題は基本的なプログラム実装力だけではなく，アルゴリズムの精度を測るようなテストケースも含まれているため，部分点のみを取得する解答者が増えているからと考えられる．

表 4. 難易度別得点分布

		難易度1	難易度2	難易度3	全体
	0	28.71%	24.04%	19.62%	25.83%
	1-9	2.19%	3.22%	3.44%	2.75%
	10-19	1.07%	6.68%	4.43%	3.72%
＜20		31.97%	33.94%	27.49%	32.30%
	20-29	1.46%	5.82%	3.86%	3.50%
	30-39	2.40%	3.39%	5.65%	3.15%
	40-49	3.62%	3.81%	4.98%	3.84%
	50-59	1.40%	3.00%	6.76%	2.63%
	60-69	1.43%	3.23%	3.70%	2.41%
≧70		57.70%	46.80%	47.56%	52.17%
	70-79	1.93%	3.87%	10.02%	3.59%
	80-89	3.29%	3.86%	7.72%	4.00%
	90-99	5.13%	9.50%	10.33%	7.47%
	100	47.35%	29.57%	19.49%	37.11%

表 5. 難易度別に見た言語選択の比率

	難易度1		難易度2		難易度3		全体	
言語	受験者比率		受験者比率		受験者比率		受験者比率	
C++	12%	51%	13%	60%	18%	64%	13%	56%
Python3	39%		47%		46%		43%	
JavaScript	4%	36%	3%	31%	2%	28%	3%	33%
Ruby	7%		6%		6%		6%	
C#	5%		3%		2%		4%	
PHP	5%		5%		2%		5%	
Python2	3%		4%		3%		3%	
Java	12%		10%		11%		11%	
C	13%	13%	9%	9%	8%	8%	11%	11%

表 4 からもう一つ読み取れることは，難易度が上がるにつれて満点の割合が減少する度合いが，ゼロ点の割合が減少する度合いより大きいことである．

しかし，表 4 によれば，70 点以上の割合と，20 点未満の割合は難易度によって大きく変化していないという，直感に反する結果になっている．その理由は，表 5 からわかるように，問題の難易度が高くなればなるほど,「人＋言語」の能力の高い C++や Python3 での解答比率が上がっているからと考えられる．これは優秀な生徒だけを集めても，その中では必ず相対的な優劣差がつくという一般的な現象と符合する．実際，難易度の高い問題を出題する企業の受験者は，ある一定以上のレベルを有している受験者に限られているとも推測される．

3.3 カテゴリ別分析

カテゴリ別の分析を表 6 にまとめた．2.4 節で定義した問題カテゴリのうち，LL，SQ，LG，SY のカテゴリの問題は track にまだ存在していないので表 6 から省いたが，track に増やすべきカテゴリとして今後の課題としたい．

それぞれのカテゴリで最も高い平均点を緑，最も低い平均点をオレンジで着色した．

その結果，どのカテゴリにおいても全体を通して C は他の言語と比較をすると成績が著しく低いことがわかる．また，最高点の中でも，C++は木とグラフ (TG)，再帰とダイナミックプログラミング (RD)，探索とソート (SS) において他を圧倒する成績を出し，Python3 が数学関連 (MA) の問題で他を圧倒する成績を出しているのは興味深い．また，C が他の言語と比較して，全体的に成績が悪く，とりわけ配列・文字列操作 (AS) の成績が著しく悪いことがわかる．

表 6. プログラミング言語・カテゴリ別平均点

	BA	AS	TG	BI	MA	RD	SS	AD
C	47.50	23.89	17.24	26.76	26.20	30.24	33.12	9.72
C#	57.73	49.34	25.00	27.96	55.35	47.51	49.72	24.86
C++	76.04	69.38	65.94	64.56	65.85	68.35	81.63	47.71
Java	53.76	42.01	24.61	30.55	46.19	40.26	48.26	33.47
JavaScript	66.15	55.48	22.19	43.85	61.48	50.53	63.85	37.77
PHP	52.23	50.93	20.13	31.70	52.37	50.91	57.01	37.18
Python2	52.30	43.75	20.78	48.95	57.26	50.57	44.83	39.72
Python3	76.75	63.67	37.57	54.96	74.78	58.35	74.44	46.89
Ruby	55.07	50.41	27.37	36.94	60.89	47.36	54.18	27.71
平均	**63.74**	**55.76**	**40.45**	**50.91**	**61.59**	**54.02**	**65.19**	**38.91**

3.4 Python2 系 vs Python3 系

今回の調査で意外だったのは，Python2 系と Python3 系の得点の差である．表 1 より，Python3 の解答数は Python2 の解答数より 10 倍以上多く，表 2 の通り，平均点は 1.44 倍高い．数千サンプルあるので，Python3 での解答の得点が有意に高いと言える．

Python2 と Python3 の言語としての違いにおいては，標準のライブラリ，演算手法や型の扱いなど，特徴差はあるものの，実行の速度やデータ構造の扱いなど，言語としての性能の差が大きいと考えるのは難しい．

また，Python2 は MacOS がデフォルトで設定しているバージョンであることから，初心者が気軽に扱う機会が多いことを考慮すると，Python2 に留まった人（バージョンアップの必要を感じなかった人）のスキルは高くなく，プログラミング経験が浅かったのではないかと推察する．

4. 考察

4.1 プログラミング言語別分析の考察

上位になった C++と Python3 で問題を解答している受験者の中から成績優良者（満点解答者）の一部に対して，事後であるが，アンケートを行った．その結果を図 5，図 6 にまとめた

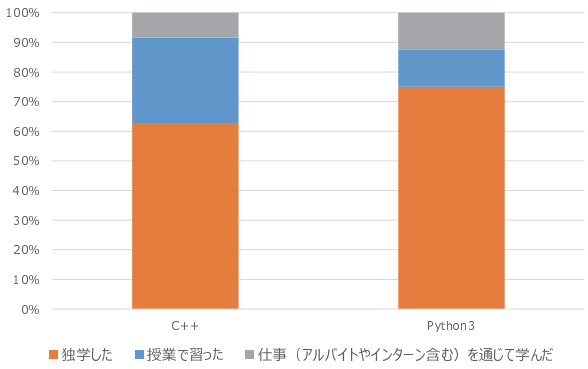

図 5　C++と Python3 の成績優良者の学習形態

まず図 5 で興味深いのは，C++と Python3 のいずれも，成績優秀者は授業等で習ったのではなく，独学で習得している比率が高いことである． C++を選択した学生は，研究室等での必要に迫られて，あるいは趣味（競技プログラミングコンテストサイトなどの練習問題）で C++を独学し，使いこなしてきたことが図 6 のアンケート結果から窺える．

Python3 は最近東京大学が入門言語として採用したことで話題になったが，独学においても入門言語として扱われる機会が多く，かつ近年盛り上がっている機械学習に推奨されている言語でもあるため，実プログラミング経験を積むチャンスが多かったことが，アンケート結果から窺える．

最下位になった C 言語はいまだに入門言語として使われているが，例えば HashMap，queue，stack 等のデータ構造が用意されておらず，アルゴリズムを実装する際はすべて自前実装しなければならない．時間制限のある試験においては，問題で問われる本質な部分以外の実装が他の言語と比較して重たく，短時間での書きやすさが他の言語よりも低いことが大きな要因として考えられる．また，track ではコーディング形式の問題の解答を開始した際のデフォルトの解答言語が C であるため，それ以外に得意な言語を知らない学生や，プログラミング経験の浅い学生の受験機会が多かったとも考えられる．

しかし，高い難易度の問題でも満点の学生がいるので，システムプログラミングなどに向いた，機械語に近い C が好きな優秀な学生も一定数いると見るべきであろう．

4.2 カテゴリ分析の考察

機械学習というホットトピックスに適合し，入門障壁の低い Python3 は，数学分野において高い成績を出しており，さらにどのカテゴリにおいても他の言語に劣らないことから（東京大学のように）入門から教えて育てる上で推奨できる言語の一つであると言える．

また，ソートと探索，木とグラフ，再帰とダイナミックプログラミングなど，高度なデータ構造やアルゴリズムの実装能力の高い学生が多く使っている C++は，入門言語のあとに，研究や実社会に直結したプログラミングのために（研究室内のプログラム資産の利用の観点からも）途中から教えるという選択が，少なくとも学生を就職という形で社会に送り出すにはベターな選択と考えられる．

4.3 関連研究について

プログラミングに関する試験の採点結果の分析としてこれほどの量のデータを扱っている研究は筆者らが検索して探した限りでは見つからなかった．中国に National Computer Rank Examination[5] という，学生のプログラミングスキルのランキングを計測するシステムがあるようだが，分析結果などは公表されていない．内容は，IPA の情報処理技術者試験にある程度の実技が加わったもののように見える．

Parker ら [6] など，入門用言語をどう選ぶか，どう学ばせるかに関する論文は多いが，実際に使うべき言語という視点の論文は多くない．Goosen ら [7] は抽象的な基準について議論しているが，具体的な言語に言及していない．研究論文ではないが，具体的なプログラミング言語の選択に関しては多数の Web 記事がある．すべてアンケート調査，あるいは著者の独自判断によるものであり，年々アップデートされている [8, 9, 10, 11, 12 など].

5. むすび

採用試験という，学生にとっては真剣勝負といえる場で採集された大量の解答データには，それなりの重みがあろう．しかも，1 社にとどまらず，多くの会社

の採用試験の解答データなので，留意すべき偏りはないと考えられる．

本論文では，問題の難易度，問題の対象カテゴリを2つの軸として，試験の解答のために選択されたプログラミング言語別の得点データを分析した．その結果，得点が

(1) C++, Python3
(2) JavaScript, Ruby, C#, PHP, Python2, Java
(3) C

という3つのクラスに明確に分かれることを明らかにした．また，カテゴリ別の得点の比較では，C++が木とグラフ，再帰とダイナミックプログラミング，探索とソートの問題で他を圧倒する成績を出し，Python3では数学関連の問題で他を圧倒する成績を出している結果となった．

これは巷間では暗黙のうちに意識されていたことを，大量のデータで裏打ちした希少な分析の事例ではないかと考える．

言語は適材適所の多様性を持ち合わせており，この結果は言語自体の能力の差を示したものではなく，あくまで主に入社試験という場で観測された「人＋言語」の能力の差の結果であると捉えるべきである．ソフトウェアを生み出す能力という広い意味で「人＋言語」の能力を「プログラミングスキル」と定義していいのかもしれない．

C++とPython3の成績上位者へのアンケート結果からも窺い知れるように，得点は，学生が大学においてどのような環境に置かれてプログラミングを行ってきたかに大きく依存する．実際にリアルなプログラミングを行うことで,「人＋言語」の意味でのプログラミングスキルが養成されたと考えることができよう．

上記のような「習うより慣れよ」は，学習する言語の選択の問題ではないという意見があるかもしれないが，必要なソフトウェアを作成するために，適切な言語を選択した結果，効率的にアウトプットを出しやすくなることは確かである．C++とPython3が1つ頭が抜けた成績になったのは，これらの言語が授業で教わった言語ではなく，研究や自分の趣味を目的とした言語として選択し，学習したからということが，アンケート結果から読み取れる．

指導する教員の立場で言えば，むしろ研究などに必要なソフトウェアを作成する環境を適切に与えることが重要なのかもしれない．これは「人＋言語」という意味でのプログラミングスキルを自分の武器にしたい学生にとっても重要である．これにはプログラミングの入門言語の選択とは異なる配慮が必要である．

上記の分析データからわかるように，現在のtrackには欠けている問題カテゴリがある．今後はこれを補いつつ，さらに就職活動年度毎に，数万件の解答データを収集して，時系列による傾向の変化などを読み取っていくことが必要であろう．また，今回は解答の得点だけを分析したが，正解さらには不正解プログラムの構造の詳細な分析も興味深い問題である．本研究をベースに分析をさらに深化させていきたい．

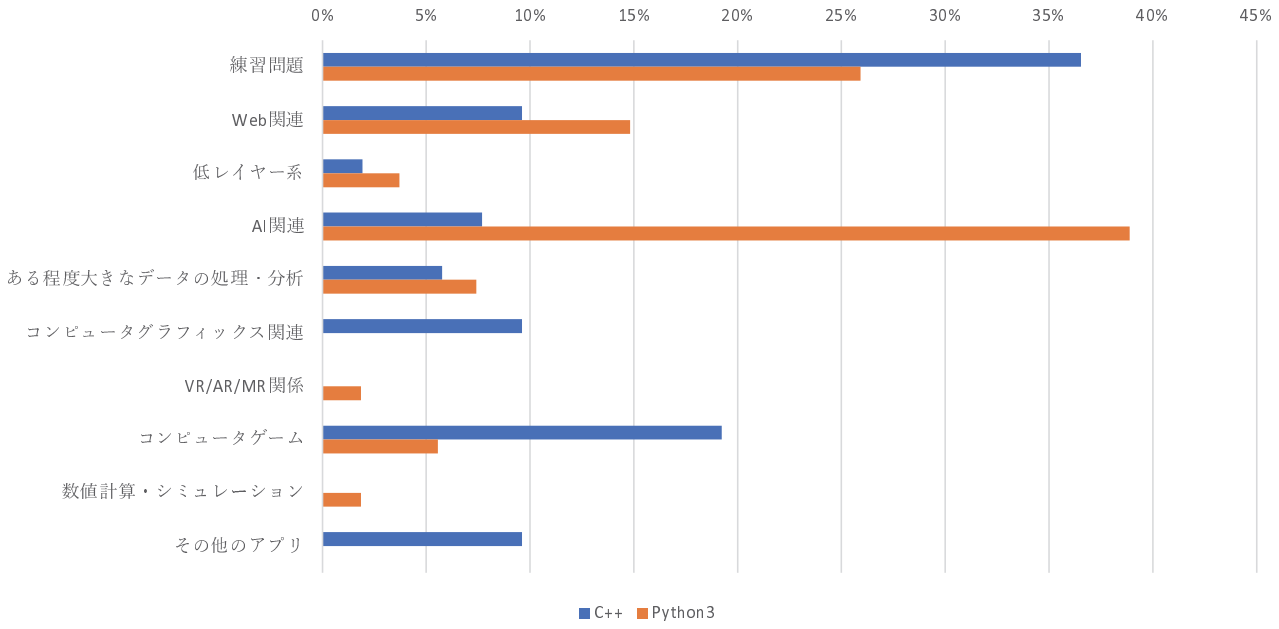

図6　C++とPython3を選択した学生のプログラミング分野

[参考文献]

[1] 岡部成玄: 一般情報教育の全国実態調査 (2)， 情報処理 Vol.56, No.1, 2015.
[2] 新田章太，小西俊司，竹内郁雄: 複数言語に対応しやすいオンラインプログラミング学習・試験システム track，情報教育シンポジウム (SSS) 2019.
[3] ヒューマンリソシア: IT 分野の市場は拡大する一方，人材は不足 https://resocia.jp/case/287
[4] Gayle Laakmann McDowell: Cracking the Coding Interview, Lightning Source Inc, 2015.
[5] https://baike.baidu.com/item/国家算机等考/934309
[6] Kevin R. Parker, et al.: A Formal Language Selection Process for Introductory Programming Courses, JITE-Research Vol. 5, No. 1, Informing Science Institute, 2006.
[7] Leila Goosen, et al.: Choosing the “Best” Programming Language, Proceedings of the 2007 Computer Science and IT Education Conference.
[8] Gaurav Belani: Programming Languages You Should Learn in 2020 https://www.computer.org/publications/tech-news/trends/programming-languages-you-should-learn-in-2020
[9] Md Kamaruzzaman: Top 10 In-Demand programming languages to learn in 2020 https://towardsdatascience.com/top-10-in-demand-programming-languages-to-learn-in-2020-4462eb7d8d3e
[10] Best Programming Languages to Learn in 2020 (for Job & Future) https://hackr.io/blog/best-programming-languages-to-learn-2020-jobs-future#: :text=1.-,Python,to%20develop%20scalable%20web%20applications.
[11] Sruthi Veeraraghavan: Best Programming Languages to Learn in 2020 https://www.simplilearn.com/best-programming-languages-start-learning-today-article
[12] The 9 Best Programming Languages to Learn in 2020 https://www.fullstackacademy.com/blog/nine-best-programming-languages-to-learn

